BEA WebLogic Workshop 8.1 ® 快速上手指南

開始使用 BEA WebLogic Workshop 8.1™



台灣比爾亞系統
台北市敦化南路二段105號23F
電話: (02) 27841623
傳真: (02) 27841625

2003年7月
BEA WebLogic Workshop™

版權宣告

使用權利說明

商標及服務標記

目錄表

如何使用這份指南

這份快速上手指南帶來一些當您準備探索和評估BEA WebLogic Workshop 8.1 ™可能是有用的重要題材。這份文件提供一個BEA WebLogic Workshop 8.1™的概觀，一個著眼在它的特色和利益的詳細觀點，一個使用BEA WebLogic Workshop 8.1™開發應用程式的討論，和當您評估這套產品時您可能需要的重要資訊。我們建議您在您開始探索這套軟體之前閱讀完這份指引。BEA已經提供了BEA一個可使您快速地和廣泛地探索BEA Weblogic Workshop 8.1™的能力的產品展示。細節的一步步指導被包括於這份快速上手指南中。

除了此份指引所包括的教材，您可能想要使用 BEA網站上豐富的線上資源，包括資料表、產品手冊、白皮書、展示軟體、新聞稿、以及其他資源。

額外的教育資源

更多的資訊、資料表、產品手冊

BEA WebLogic Workshop 產品網站
http://www.bea.com/framework.jsp?CNT=index.htm&FP=/content/products/workshop

評估軟體

BEA WebLogic Workshop 8.1™
http://commerce.bea.com/showproduct.jsp?family=WLW&major=8.1&minor=-1

新聞稿

BEA 及產品相關新聞
www.beasys.com/press/room.shtml

開發者資源

dev2dev Online, BEA 開發者網路社群

http://dev2dev.bea.com

BEA WebLogic Workshop™ 相關新聞群組列表
www.beasys.com/support/newsgroup.shtml

BEA WEBLOGIC WORKSHOP 8.1™產品概述

BEA WebLogic Workshop 8.1 爲一具有一致性、簡化的、可延伸的發展環境。由於上述特徵，使得 BEA WebLogic Workshop 8.1 讓使用者能夠非常容易地在整個 BEA WebLogic 平台上開發企業級、以標準基礎的應用程式。

BEA WebLogic Workshop 獨特的程式模型，提供了簡化的抽象概念以加速軟體的開發。抽象概念相對於傳統程式開發方法之不同，在於其目的是讓使用者能夠以更快的速度，開發更佳的應用程式。Workshop 創新的程式開發模型包括了以下三個主要元件：視覺化開發環境 (visual development environment), 執行期框架(run-time framewor)以及 Java 控制項(Java Controls).

視覺化開發環境 (Visual Development Environment)

WebLogic Workshop 的視覺化開發環境，爲 J2EE 應用程式發展提供了前所未有的易用性。Design View 的部分係以圖形化的方式，提供了開發中之應用程式的描述，因此開發者便能夠察看應用程式與客端及 back-end resources(後端資源)之間的互動，並以視覺化的方式加以編輯。此外，所有的 WebLogic Enterprise Platform 8.1 應用程式都共用相同的視覺化開發環境，能夠大幅縮減對客製控制項(custom controls)、Web services、Web applications、portals 以及 integration applications 之建置、測試及除錯的學習曲線。

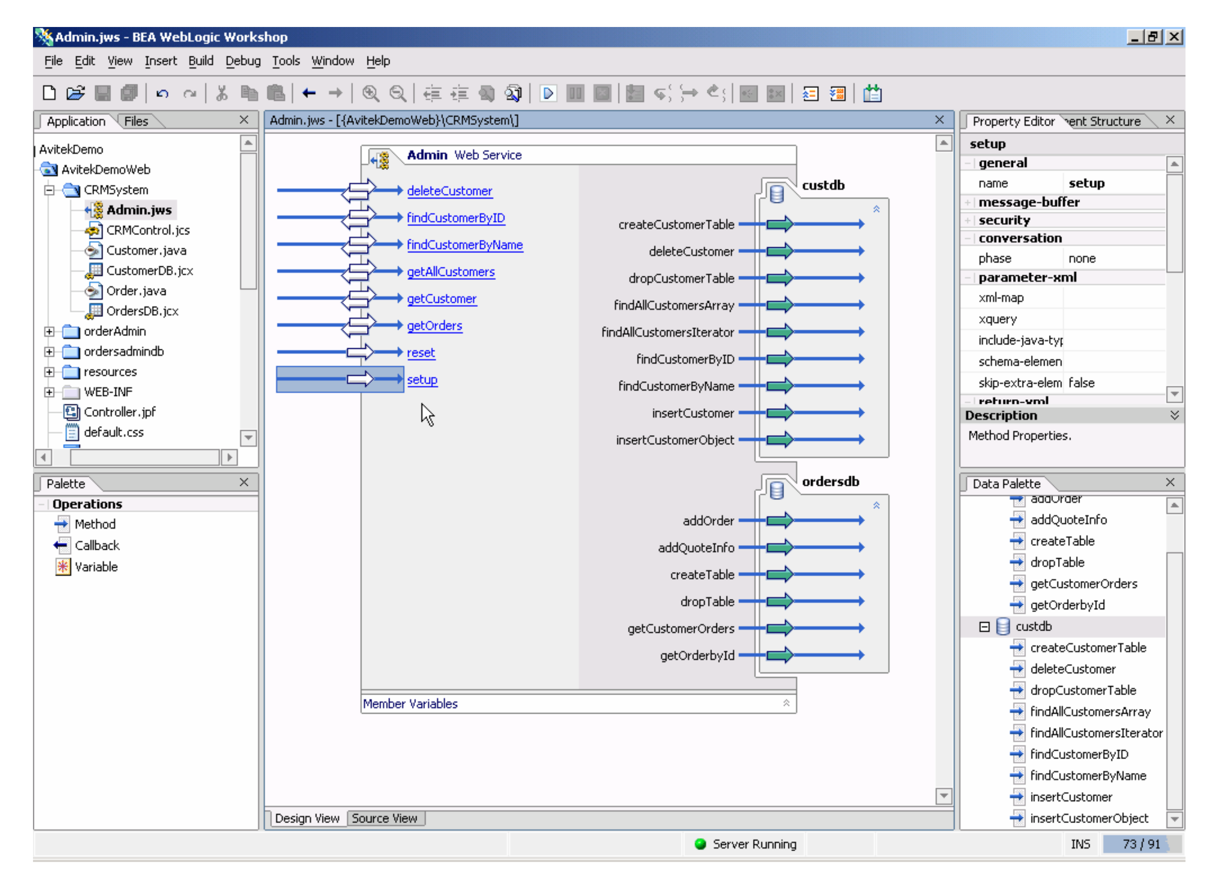


圖 1. BEA WebLogic Workshop 8.1 視覺化開發環境

WebLogic Workshop 程式開發模式係以直覺化概念為基礎。透過如同：控制項(controls), 方法(methods), 以及屬性(properties)的直覺化概念，便能夠進行以事件為基礎的程式開發(event-based development)，如此一來開發者便無須精通複雜的 J2EE API 以及物件導線程式設計的準則。此一簡化開發模式的底層機制，係使用附有註解的 Java 程式碼撰寫而成。WebLogic Workshop 視覺化開發環境產生標準的 Java 檔案時，Workshop 會適時地(例如當開發者設定屬性或增加控制項時)幫開發者加入對應於特定執行時應用程式行為的註解。這些註解可讓 Workshop 執行期框架 (run-time framework)自動產生 J2EE 的基礎結構元件，以抽象的方式讓開發者跳脫低階的基礎結構配線過程。開發者亦可自行撰寫商務邏輯，以解決手邊的商務問題。

開發者係透過 Source View 來撰寫 Java 程式。開發者能夠直接存取原始碼與原始碼註解，雙向程式編輯的特性可保證在 Design View 中所做的任何一項變動，都能夠被迅速地反應到 Source View 中，反之亦然。Source View 允許使用者撰寫程序性的 Java 程式來處理方法、事件，並提供許多提升生產力的特殊功能來協助開發者，如：程式自動補完 (code-completion)、語法驗證 (syntax checking)及自動匯入(auto-import)。

orderEntryService.jws* - [{AvitekDemoWeb}\]

```
import weblogic.jws.control.JwsContext;
import org.openuri.easypo.PurchaseOrderDocument;
import org.openuri.easypo.PurchaseOrder;
import CRMSystem.Customer;
import CRMSystem.Order;

/**
 * @common:xmlns namespace="http://openuri.org/easypo" prefix="ns0"
 */
public class orderEntryService
{
    public Callback callback;

    /**
     * @common:control
     */
    private CRMSystem.CRMControl cRMControl;

    /** @common:context */
    JwsContext context;

    /**
     * @common:operation
     * @common:message-buffer enable="true"
     * @jws:conversation phase="start"
     */
    public void createOrderAsynch(PurchaseOrderDocument podoc)
    {
        Customer c = new Customer();
        Order o = new Order();
        PurchaseOrder po = podoc.getPurchaseOrder();
        c.name = po.getCustomer().getFirstname() + " " + po.getCustomer().getLastname();
        c.custId = po.getCustomer().getCustomerNumber().intValue();
        o.custID = c.custId;
        o.orderID = po.getOrderData().getOrderNumber().intValue();
        o.cameraQty = po.getOrderData().getCameraOrder().getQuantity();
        o.flatScreenQty = po.getOrderData().getTvOrder().get
        cRMControl.createOrderAsynch(c,o);
                                        getClass()          Class
                                        getDescription()    String
    }                                   getPrice()          int
                                        getQuantity()       int
    public interface Callback
```

Design View | Source View

圖 2. 在 Source View 中協助開發者的程式自動補完功能

Workshop 的另一個特點為其具高度可靠性、支援多國語系的整合除錯器，能夠允許使用者能夠在同樣的視覺開發環境中，以簡單的方式建置、測試程式並進行除錯。BEA WebLogic Workshop

8.1 除錯器能夠允許使用者在應用程式伺服器上，針對已部署的程式碼而非本地端的 Java 檔案進行除錯，以實現即時回饋，使應用程式的品質更為提高。

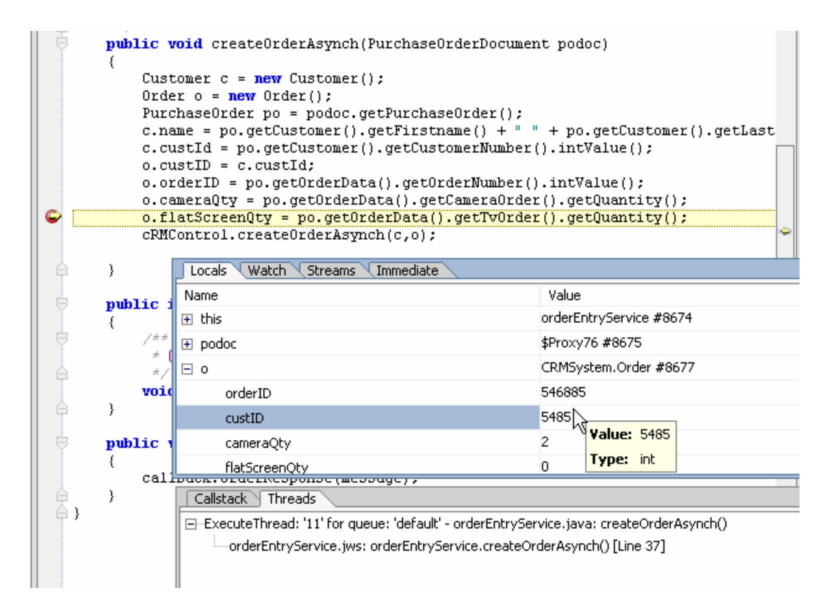


圖 3: WebLogic Workshop 的整合除錯器運行畫面

WebLogic Workshop 8.1 更進一步地提供了許多整合開發環境的加強功能，以了改善使用者的開發經驗。如：整合原始碼管理、強化的視窗化環境、完整的工具自訂組態能力，以及共享的工具及調色盤。

執行期框架 (The Run-time framework)

BEA WebLogic Workshop 的 執行期框架提供了介於開發者與複雜 J2EE 系統基礎結構之間的抽象層。開發者可免於處理耗時的、API 層次的基礎結構程式碼、元件組態及部署的細節，而自由地使用視覺化的開發環境，在必要的部分撰寫程序性的 Java 程式碼，並以簡明、宣告的註解存取進階的功能。WebLogic Workshop 執行期框架便是透過上述方式，產生標準的 EJB, JMS 以及 JDBC 程式碼，並以此程序管理與 J2EE 結構相關的設計及實作要點，故其應用程式之實做，皆立基於一可靠的、具規模度且安全的企業級架構之上。

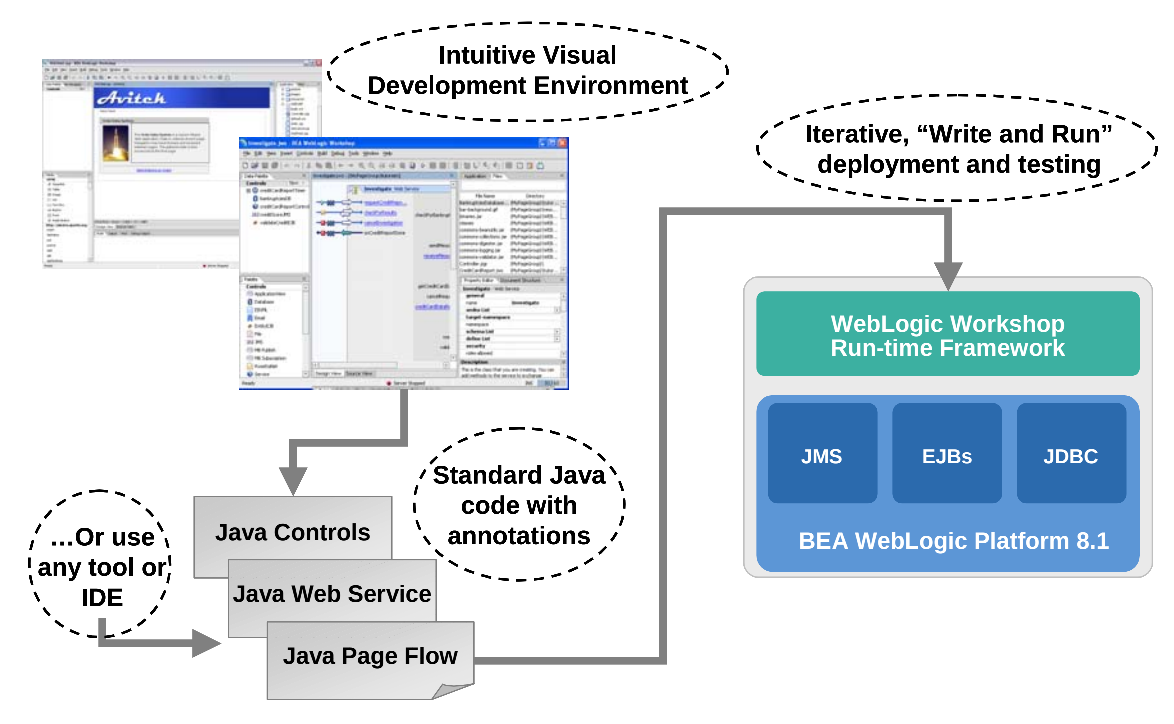


圖 4. 連結設計時期(Design-Time)與執行時期(Run-Time)元件

WebLogic Workshop 的執行期框架為一執行於 BEA WebLogic Server 8.1 上的標準的 J2EE 應用程式，它代表著統合所有 WebLogic Platform 應用程式型別(type)的聚合層。

BEA WebLogic Workshop™的執行期框架利用抽象的概念，透過下列方式幫助開發者由開發及部署企業級應用程式的複雜細節中跳脫出來：

- ***支援 Java Web Service (JWS)檔案：*** WebLogic Workshop 於 7.0 版時便已採用 JWS 檔案的方式開發 Web Service 應用程式。JWS 即為 Java 檔案及一組 Javadoc 註解，開發者可透過 JWS 指定屬性(properties)，並透過簡明宣告註解(simple declarative annotations)的方式，獲得關鍵 Web service 功能之存取權。透過此一方式，開發者便無須著力於 SOAP marshalling、Java-to-XML binding, 建立 WSDL 檔案、底層 bean 之部署等細節。JWS 目前正透過 JSR(Java Specification Request) 181 進行標準化中。

- ***支援 Java Page Flow (JPF)檔案：*** WebLogic Workshop 8.1 的 Web 應用程式係以 Java Page Flow(JPF)建置而成，JPF 是一項創新的 Java 技術，允許開發者以更簡易的方法，透過視覺化的方式設定頁面的處理流程，以建置企業級的網站應用程式。開發者於運作中撰寫的商務邏輯，可由應用程式中的頁面直接叫用。為了大幅簡化資料的存取機制，WebLogic Workshop 提供了 tag libraries(標籤庫)及拖放精靈，可經由任何一個資料庫、Web Service 或 Java Control 自動進行頁面及資料元素(data elements，如：field, list, tree, grid 等)的資料連結(data-binding)。此外，執行期框架會自動支援 session 及 statemanagement(狀態管理)等細部功能，最後在底層運用 Struts 的 MVC(model-view-controller)架構上，產生 Java Page Flow 的應用程式。

- ***支援 Java Controls (JCX/JCS)檔案：*** Java 控制項(Java Controls)可讓開發者透過一個已經為許多開發者所熟悉的簡化介面，存取後端的企業資源。開發者可透過實做方法(method)的方式，處理事件以存取資產(如：資料庫、外部 web service 及 EJB 元件)中的資訊。透過設定屬性的方式，即可設製符合個別需要的控制項。執行期框架亦允許自製 Java 控制項的彈性，開發者可自行撰寫符合其別需要的 Java 控制項。透過控制項可建置功能強大的非同步商務邏輯，所建

置之商務邏輯亦可於交談式的 web service 及工作流程(Business Process)中叫用。詳情請見下一節。

- ***Portal/Integration 應用系統延伸:*** WebLogic Workshop 8.1 Platform Edition 包含了允許使用者個別結合 BEA WebLogic Portal 或 BEA WebLogic Integration，以建置、測試及部署進階入口網站(Portal)及 Workflow/EAI/BPM 應用程式的框架延伸。Workshop 執行期框架提供了一個跨越上述所有應用程式型別的整合架構，能賦予開發者無接縫的開發經驗。

- *支援非同步通訊：*撰寫非同步通訊及雙向傳訊的管線程式碼是非常累人且耗時的。WebLogic Workshop 將此程序予以簡化：藉由簡易的屬性設定，可開啓訊息緩衝功能(message buffering)，自動產生對 Web service 及自訂控制項的非同步支援。使用者可由 J2EE Messaging API 中跳脫出來，由框架來處理佇列管理的細節。在 Web service 的部分，Workshop 爲主客端間的訊息交換提供了一個簡易的交談式暗喻, 框架會自動連結相關訊息，並對長期會談之狀態進行管理。

- ***XQuery Mapping 及 XMLBeans:*** Workshop 提供了許多解決方案以確保應用程式間的真正的低藕合度，以提升應用程式的可靠度及規模度。XQuery Mapping 功能允許開發者以視覺的方式設置對應的資料欄位，執行期框架會自動處理複雜的底層資料轉換。XMLBeans 也提供了有型別(Strongly-typed)的 Java 物件介面，保持了對底層 XML 訊息的完整存取能力。結合以上 WebLogic Workshop 執行期框架支援的解決方案，能夠確保 XML 資料永遠受到「頭等市民」般的對待。

- ***邊寫邊執行測試的部署:*** Workshop 在執行期框架支援一觸自動部署至 BEA WebLogic Platform 的過程中，提供強化的遞迴開發 moel。如此一來，開發者得以將焦點放在確保程式碼的品質上，而無須顧慮底層的平台，不但可大幅縮短建置、測試之週期，亦可顯著提升生產力。

- ***自動產生測試模擬環境:*** 執行期框架會自動爲每一個部署好的應用程式，產生一個模擬瀏覽器運作的測試客端程式，以確認正確的功能，消除測試週期中耗費成本的瓶頸。此一功能讓開發者可以快速地撰寫程式碼的步驟並測試其正確性。

Java 控制項 (Java Controls)

在 WebLogic Workshop 8.1 中，Java 控制項讓開發者可以輕易地連結現有的資料、系統、應用程式以及商務邏輯。Java 控制項爲附有方法及屬性的視覺化元件，可以處理與外部資源或商務邏輯連結的所有相關 J2EE 細節。開發者透過事件處理及屬性設定與控制項進行互動。Workshop 的其他部分亦提供類似的、簡化的 Java 程式方法，可以很容易地建置出訂製的控制項。上述特徵讓開發者可宣告式地指定行爲，並進而把心力放在使用標準程序性 Java 程式碼以進行事件處理及方法呼叫上，無須學習多餘的 API。此外，Java 控制項亦提供存取資源及應用程式邏輯的 best-practices，以確保資源使用的最佳化。

BEA WebLogic Workshop 8.1 包含了下列 Java 控制項：

- ***Web Service 控制項 -*** 讓開發者可以用存取 local 物件的方式來存取 Web services，而無須顧慮任何低階的 Web service 通訊協定細節。開發者只需爲目標服務指定 WSDL 檔案，控制項便會自動設置好連線，並在視覺化開發環境中自動顯示出目標服務的方法。Service 控制項使連結 Web service 變得更加容易，開發者無須顧慮實做的技術細節，便可輕易地存取包含以.NET 爲基礎的服務。

- ***Database 控制項 -*** 讓開發者可以把心力放在需要從資料庫取出的資料上，而無須顧慮 JDBC(Java Database Connectivity)API。開發者可以集中精神於他們所熟知的部分(如 SQL 陳述式及 Java 函式)，Workshop 的執行期框架會自動處理底層的管線配置(plumbing)。

- ***EJB 控制項* -** 可支援團隊開發模式。應用程式開發者所使用的商務邏輯，可能已經存在，或正由企業開發者開發中。EJB 控制項可簡化 EJB 程式開發模型，使遠端的 EJB 物件以 class 的形式出現，讓開發者進行屬性設定及方法呼叫。

- ***The JMS 控制項* -** 讓開發者可以輕易地存取訊息佇列，而無須顧慮 Java Messaging Service API 的細節。開發者可以把焦點放在來往訊息的內容上，而無須顧慮其間的機械化結構。

- ***The Timer 控制項* -** 可幫助開發者撰寫非同步程式。Timer 控制項可用來實做 message timeouts、以 poll 的方式決定 web service 進行的完成度，並協調多重非同步的訊息回應。

- *其他*：WebLogic Workshop 8.1 增加了許多內建的 Java 控制項，讓開發者可以輕易地連結並使用 IT 資產，如：FTP, eMail, Tuxedo, Portal, Integration 控制項等。

除了上述內建 Java 控制項外，BEA WebLogic Workshop 8.1 也允許自行實作 Java 控制項的彈性，使用者(包括 ISV)可自行建置訂製的 Java 控制項，將其無接縫地插入 Workshop 發展環境中，使其成爲具高度再用性的商務邏輯元件。開發者如欲建立訂製的 Java 控制項，只需使用已熟悉之視覺化設計工具以指明介面(例如：支援的方法及事件)，設定用於描述執行時期行爲的屬性，再使用實作這些方法的程序性 Java 程式碼來撰寫商務邏輯即可。

Java 控制項可延伸的本質，使其成爲部署服務導向(service oriented)架構的理想模式。在服務導向的架構中，商務邏輯元件以獨立模組的形式產生，可被重複使用以服務多種終端應用程式。使用 Java 控制項來實作此一有效、具再用性的模組化設計具有下列優點：

- Java 控制項在 design-time 時即擁有自我描述的能力，如此一來開發者便能夠在 Workshop 的開發環境中，獲取伺服器端商務邏輯的視覺化表示。

- 使用者可透過 Java 控制項提供的一組簡易屬性，實做進階的執行時期功能如：非同步通訊、安控角色, 生命週期事件管理以及交易的支援等。此外，開發者尙可利用列示控制項屬性表的功能，以利後續使用者的使用。

- Java 控制項可以標準 JAR 檔案的形式輕易地封裝並散佈。此外，Java 控制項本身可巢狀配置的特性，使其成爲易於再用的元件。

- 開發者可透過 Java 控制項的抽象概念，輕易地連結至任何的 IT 資產、ISV 應用程式或商務邏輯，而無須熟悉大量的基礎架構複雜程序。開發者可將心力放在程序性 Java 程式而非 J2EE 或其他廠商專屬的 API 上。

再用元件已成爲所有 WebLogic 平台應用程式的中央構件。而 Java 控制項則爲一將商務邏輯封裝爲再用元件的絕佳方式。

BEA WebLogic Workshop 8.1 之應用程式型別

視覺化開發環境, 執行期框架及 Java 控制項三者促成了 BEA WebLogic Workshop 的簡化程式撰寫模式。重點在於此一創新的開發模式，具有高度的一致性，所以開發者只需學習一種程式撰寫模式，便能夠建置並整合整套 WebLogic Platform 的應用，如：Web services, Web 應用程式, Portals 及整合應用程式。

企業級 **Web Services**

企業的 IT 專業人士認知到 Web services 能夠減少基礎結構的複雜度，並有效控管以往跨越不同系統及地理區域，進行整合等應用之相關成本，具有龐大的商業價值。然而如欲成功，就必須跳脫出今日普遍爲人所熟知的簡易、同步的 Web services 觀點。BEA WebLogic Workshop 8.1 在 Workshop 於 Web service 的強力領導之上進一步建制及延伸，爲企業級的 Web service 提供了內

建的結構性支援，具有非同步、低藕合、商用 XML 文件、更高的安全性及可靠的訊息傳遞等特點。對於建置能夠有效地克服企業挑戰(如：應用程式整合)的 web service 結構而言，這些特點是相當關鍵且重要的。此外，WebLogic Workshop 提供了 XQuery 對應以及 XMLBeans 技術，兩項突破性的創新發明可將讓 Java 程式開發者處理以 XML 爲基礎的資料及文件時，更具生產力。

您將於範例三中學習到如何產生一個企業級的 Web Service，並於範例六學習如何使用 XMLBeans。

強大 Web Applications 開發環境

BEA WebLogic Workshop 8.1 引進了一組視覺化設計工具, 控制項及可延伸框架讓開發者使用動態 JSP/HTML 使用者介面，產生功能強大的伺服器端應用程式。Workshop 提供拖拉放的視覺化編輯器，以協助開發者建置動態的、資料連結的頁面；此外，尚可透過 Java Page Flow 技術輕易地運用強大的應用程式功能。Java Page Flow 提供了一個協助開發者組合企業級 Web 應用程式的抽象層。藉由指定頁面、動作、導覽及資料，完全不須顧慮如：sessions、狀態管理及資料連結(data-binding)等複雜的細節。Page Flows 建構於支援 MVC 的強固架構之上。其實際運作方式，則是將 Page Flows 編譯爲標準的 Java 程式，並於公開原始碼的 Struts framework 上執行。BEA WebLogic Workshop 8.1 讓開發者透過簡化的視覺化程式撰寫模型，建置複雜的、多重狀態的、資料豐富的 Web 應用程式，結合了企業邏輯及呈現邏輯，並運用了一般的 Java 控制項 methodology。

您將於範例四中完成一個 Web 應用程式，並於範例五直接從 Java 控制項中自動產生一個新的 Web 應用程式。

Portals 與 Integration 應用

傳統上在大部分的 IT 環境中，開發者都會遭遇到異質性的工具及環境，建置並整合相對於 Portal、Workflow 等應用程式之個別訂製的應用程式時，不同的工具及配套技巧是必須的。效率不彰導致開發者的生產力驟減，並提高了基礎架構的複雜度。而現今 WebLogic Workshop Platform 8.1 以允許開發者結合先進 EAI/BPM 的功能領導業界，並以單一環境、使用相同程式撰寫模型以統合 workflow、呈現及商務邏輯的方式，將 portal 功能納入訂製的開發計畫。爲此 WebLogic Workshop 8.1 Platform Edition 包含了框架延伸、額外的控制項以及視覺化設計工具以賦予開發者使用 WebLogic Portal 及 WebLogic Integration 分別建置並整合所有 Portal 應用程式及整合應用程式的能力。

此導引文件中所提供之產品描述，係以 WebLogic Workshop Application Developer Edition 所支援的功能爲基礎，不包含以 WebLogic Portal 及 WebLogic Integration 功能爲主的範例。

BEA WEBLOGIC WORKSHOP 8.1 快速上手範例

這篇快速上手指南將會引導您應用 BEA WebLogic Workshop™ 8.1 建置企業級的應用程式。經由一步一步地指引，您將會更了解 Java 控制項，以及知道如何使用它們來存取關聯式資料庫。您也可以建置客制化的 Java 控制項 和眾資料庫作非同步的整合。最後，您將可以利用這個控制項的技術來建置企業級的 Web service 和 Web application。

在實做快速上手 的範例時，您將會體驗到 Workshop 的視覺化開發環境以及簡易、共享式的程式設計模組。再者，您將受惠於 Workshop 的執行架構及簡化的 J2EE 開發和部署的能力。只要您把快速上手指南上的範例完整體驗一次，就可以在短時間內完成一個完整的開發流程—建置應用程式、整合外部資源、部署企業認可的應用架構，以及完成即時測試。

若沒有 BEA WebLogic Workshop 8.1 的支援來建置類似上述的應用程式，您至少花上數日才能完成。但倘若您使用 BEA WebLogic Workshop 8.1，則短短一兩個小時內即可搞定！

安裝 BEA WebLogic Workshop™

快速上手指南中的範本需要您安裝 WebLogic Workshop 8.1 以及 JumpStart Kit。您將會需要以下元件：

- BEA WebLogic Workshop 8.1—您可以利用CD安裝或是從以下網址下載安裝：http://commerce.bea.com/showproduct.jsp?family=WLW&major=8.1&minor=0
- 快速上手套件 (BEA_WebLogicWorkshop81_JumpStartKit.zip)—裡面包含這份教學手冊及範例所需要的專案檔。這份zip檔可由BEA WebLogic Workshop 的網址下載：http://www.bea.com/framework.jsp?CNT=step_2.htm&FP=/content/products/new_releases/workshop/get_started

當您準備好了這些檔案，請依照下列步驟安裝 BEA WebLogic Workshop 8.1 及準備快速上手的範例：

- 雙擊執行 platform811_win32.exe。這將會啓動 BEA WebLogic Platform 8.1 的安裝程式來安裝 BEA WebLogic Workshop 8.1™。
- 請按下 Next ，越過歡迎畫面。
- 請閱讀並同意 License Agreement，接著選擇 Next 繼續下一步。
- 請確定您是否要將 BEA WebLogic Workshop 8.1™安裝到預設的 *c:\bea*，確認後按下 Next 進到下一步。
- 按下預設選項的“Typical Installation”(典型安裝)，並點選 Next
- 當您確定產品安裝目錄爲預設的 *C:\bea\weblogic81b*，請選擇 Next 到下一步。
- 請按下 Next 開始執行安裝，安裝程序將持續約 15 分鐘。
- 在安裝完成的對話視窗中，請取消各核取方塊中的選項(這裡不需要安裝 XMLSpy 或是使用建置精靈)，並按下 Done，表示安裝完成。
- 在 BEA WebLogic Workshop 8.1™安裝完成後，接著準備範例所需的專案檔案。請將 *BEA_WebLogicWorkshop81_JumpStartKit.zip* 解壓縮至路徑名爲 *c:\workshop81demo* 的新資料匣。這個動作完成後您的 *c:\workshop81demo* 資料匣中應含以下三個檔案：

- o BEA_WebLogicWorkshop81_JumpStartGuide.doc (本教學手冊)
- o demo-orderentry.zip (應用程式樣本—您毋需再解壓縮此檔案！)
- o purchase-order.xsd (本範例教學中所用到的 xml schema)

- 在範例教學過程中您將會用到這些檔案，安裝完就準備好可以開始試用範例了。

快速上手範例情境－**Avitek Electronics** 公司

這整個範例是以一家叫作 Avitek 的虛擬公司爲主題。Avitek 主要營業的項目爲大眾消費性電子產品，它有一個自己發展的專屬訂貨管理系統。如今只有少數被授權的配銷商可以直接使用資訊系統向 Avitek 下訂單，其他種類訂單仍然是人工處理。Avitek 保有一些維持公司運作必要的資料庫及後端運算資源。任何企圖想擴張現有管理系統的動作，都必須從以往 20 年個別開發的 IT 架構開始實作，但必須和老舊系統整合的情形使得想要使系統現代化的理想變得太複雜且不敷成本。

最近，許多 Avitek 主要的競爭對手引進了新的配銷管道。爲了維持在業界的競爭力，Avitek 必須相對地在增加配銷管道上作出回應。首先，Avitek 希望所有的配銷商都可以利用標準的 Web services 輕易地連上 Avitek 的訂單輸入系統(Order Entry System)。接下來，Avitek 希望可以建置一個自己的網站讓消費者可以直接在線上購買某些產品，藉以掌握消費者對電子商務與日俱增的需求。爲了能成功，Avitek 必須保證這些商業應用有下列的特質：

- 利用標準化來確保大量配銷商之間整合運作不會出問題。
- 保持足夠的彈性，以允許系統作必要的變更，以及與合夥廠商間隨時作資訊的更新。
- 和既有維持訂單管理程序運作必要的資料、系統、應用程式及商業邏輯作整合。
- 搭配處理舊系統及外部資料來源間的問題，這些商業應用程式的設計也必須兼顧調節系統潛在部分和外層服務間有出入的地方。
- 這些商業應用程式必須可靠、容易取得及掌握，方能滿足擴展使用者基礎及維持在業界中品質形象的需求。

爲了滿足這些需求，Avitek 決定使用 BEA WebLogic Workshop 8.1 來協助解決眼前棘手的問題，並改善專案計畫的價值時程(time-to-value)。Avitek 計劃中的 Order Entry System 如**圖 5** 所示。

由於 Avitek 需要快速建置及擴展其新式的 Order Entry System，而公司中許多主力 J2EE 工程師早已忙於維護目前系統核心，故 Avitek 計劃委派公司其他既有的應用程式開發人員來建置新系統。即使這些開發人員中許多人不熟悉 J2EE，但所選擇的開發平臺仍必須要能爲這些開發人員所接受，同時也要能兼顧 Avitek 合夥廠商及顧客所需要的可靠性、易使用及高度的控制性。在這篇教學手冊中，您將會建置 Avitek Order Entry System 來幫助 Avitek 維持和擴展其在於競爭激烈電子市場中的占有率。

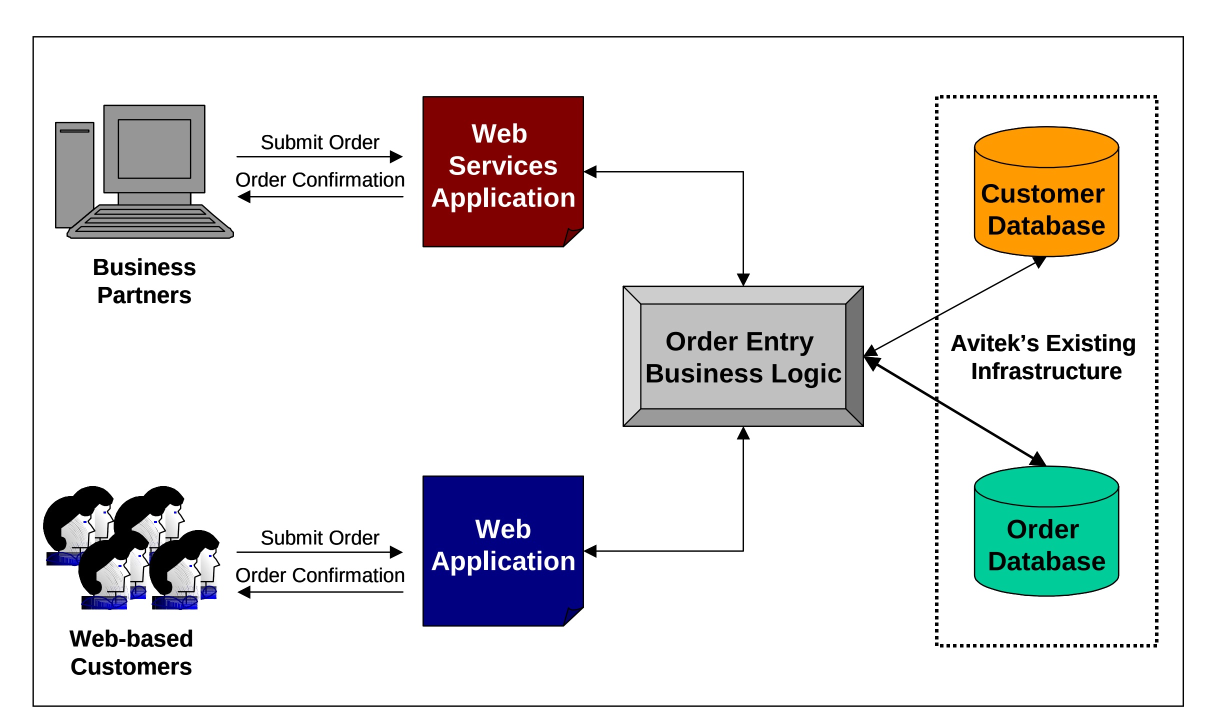


圖 5. Avitek 計劃中 Order Entry System 的架構圖

一步步體驗 BEA WebLogic Workshop™

本篇快速上手指南將會帶您走過建置企業級應用 Order Entry System 的每一步，來改善 Avitek 的商業效率、顧客反應及市場表現。

範例 1 中，您將會熟悉 Workshop 的開發環境，並學會建置工作環境，為之後的範例打好基礎。

範例 2 中，您將會建置一個客制化的 Java 項，可以非同步地和 Avitek Customer Database 及 Avitek Order Database 作互動，並可實作出訂單管理必要的商業邏輯。

範例 3 中，您將會利用之前所建客制化的控制項產生一個非同步、安全及鬆散耦合的企業級 Web service，藉此讓 Avitek 可以迅速地對外部成千上萬的潛在合夥廠商使用更進步 Order Entry 的功能。

範例 4 中，您將會再度使用到客制化的控制(custom control)，但不同的是這次是用動態的 JSP/HTML 使用者介面來建置、測試及部署一個網路應用程式。因此可以讓 Avitek 有效地和數以百萬計、願意直接線上交易的潛在消費者直接面對面地接觸。

若您還想更深入探索 BEA WebLogic Workshop，我們有另外替您準備兩個附加的範例，讓您體驗本產品不同的功能。範例 5 展示由 Java 項 從零開始快速建置的 Web application－管理 Avitek Order Database 的應用程式。範例 6 則是示範如何應用 XMLBeans 的強大功能管理 BEA WebLogic Workshop 8.1 應用程式產生的 XML 資料及文件。

範例 #1. 導覽與設定範例資料

某種程度上，為了能讓您輕鬆體驗 BEA WebLogic Workshop 8.1 的特性與功能，我們在這份快速上手套件中提供幾個已經定義好的應用程式元件，這些應用程式元件是延伸於新版 Avitek Order Entry System。

- 若要使用導覽的 Order Entry System 樣版，請從 *c:\workshop81demo* 資料夾複製 *demo-orderentry.zip* 這個檔案到 *C:\bea\weblogic81b\workshop\templates* 資料夾。注意：不需要將 *demo-orderentry.zip* 檔案解壓縮，只需要完整複製該檔案。假如你安裝 BEA WebLogic Workshop 的應用程式路徑不是 *c:\bea*，請自行調整放置的路徑。

現在準備開始探索 BEA WebLogic Workshop 8.1：

- 根據以下路徑開啓 Windows 開始功能表來啓動 BEA WebLogic Workshop 8.1：**「開始」功能表→程式集 →BEA WebLogic Platform 8.1 → WebLogic Workshop 8.1**
- 開啓 BEA WebLogic Workshop 之後會顯示一個預設的應用程式，然而你需要新建一個 Avitek Order Entry demo 的應用程式。
    - 首先，關閉所有已開啓的 Project，請選擇 **File → Close Application**。
    - 利用選單，選擇 **File → New → Application**，當顯示 *New Application* 對話框時，請選擇右邊面版的 *Demo: Order Empty Demo Application* 選項，並且輸入一個應用程式名稱，如：*AvitekDemo*。
    - 在 Server 這個欄位，確認 *c:\bea\weblogic81b\samples\workshop* 是這個應用程式的 domain，如同**圖 6** 所示。
    - 按下 **Create**，在之後出現的對話框按下 **Yes**，以確定你想要新建一個新資料夾。

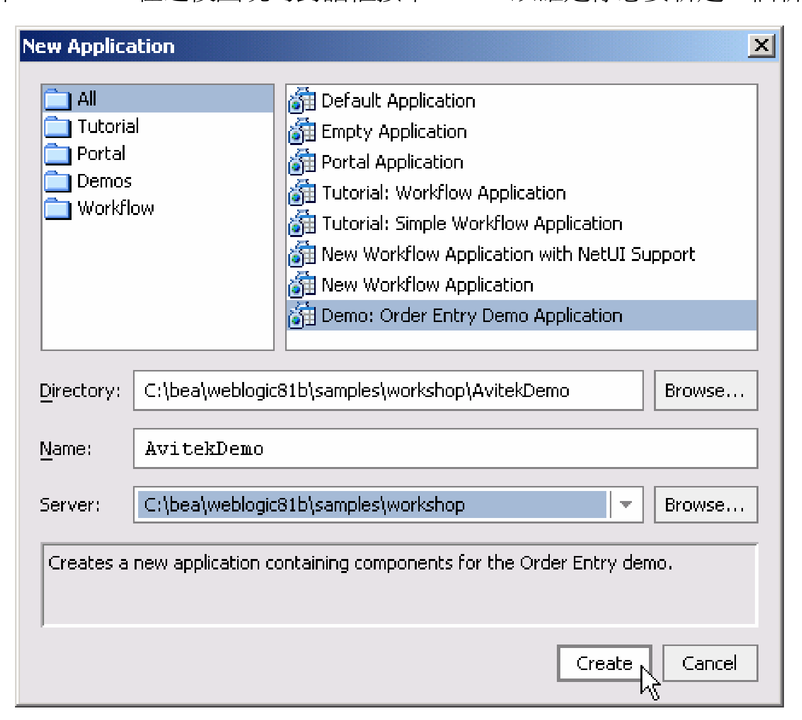


圖 6: 新建一個以 Order Entry Demo 樣版為基礎的 Workshop 應用程式

- 現在以 Order Entry Demo template 為範本的 *AvitekDemo* 應用程式已經新建完成，透過瀏覽左上方的 Application 視窗，你可以開啓這個應用程式底下的所有相關檔案。
- 接下來，展開 *AvitekDemoWeb* 專案樹狀結構底下的 *CRMSystem* 資料夾，然後按兩下 *Admin.jws* 這個檔案。這個動作在 Design View 之下開啓 *Admin.jws* 這個 web service 檔案。

視覺化開發環境

接下來該介紹 WebLogic Workshop 8.1 視覺化開發環境，如同**圖 7.**所示。這個開發環境包含一些符合你所開啓的應用程式類別而顯示的視覺化設計。假如你已經照之前已經開啓了 *Admin.jws*，你將會看到可以開發 Web service 應用程式的 JWS 設計工具。在畫面的中間是 Design View，它提供開發者一個 Web service 的圖形。這個圖形的左邊箭頭顯示了可提供客戶端的操作，而透過圖形右邊箭頭連結後端系統資源和企業邏輯。Design View 是由一些可調整的視窗圍住來幫助開發者進行工作。舉例來說，Application Window 提供容易存取及檢視正在使用中應用程式的資源；Property Editor 能讓開發者快速地設定控制項與方法的屬性。

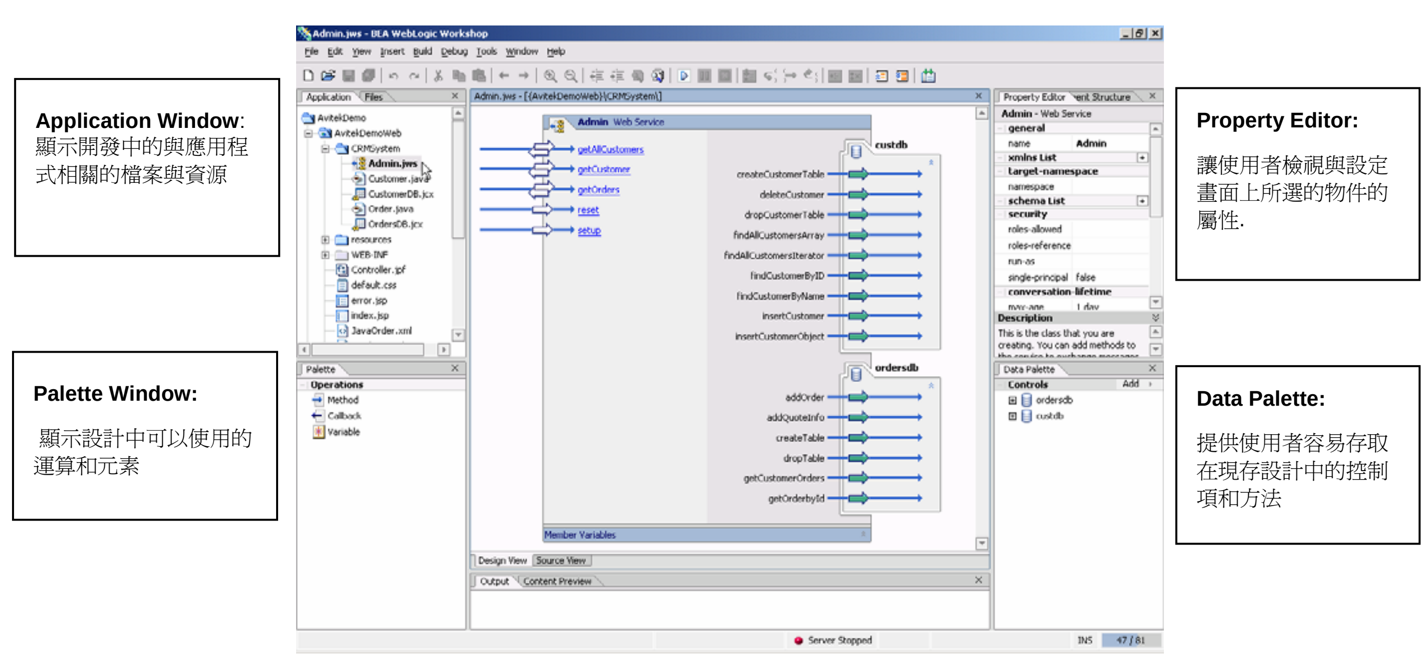


圖 7. BEA WebLogic Workshop 8.1 視覺化開發環境

Avitek Order Entry System 需要整合兩個資料庫：一個是客戶的，另一個是訂單的。為了達到這個目的，你需要先確認資料庫欄位已經正確的建立了。為了方便，設定資料庫的方法已經透過 *Admin.jws* Web service 來提供了。

- 想要建立資料庫並載入測試資料，你必須透過選擇功能選單的 **Debug → Start** 或按下工具列的 **Start** 圖示來執行 *Admin.jws*。另外也可以選擇 **Debug → Start without Debugging** 會有較高的性能，因為在此時還不需要除錯器。

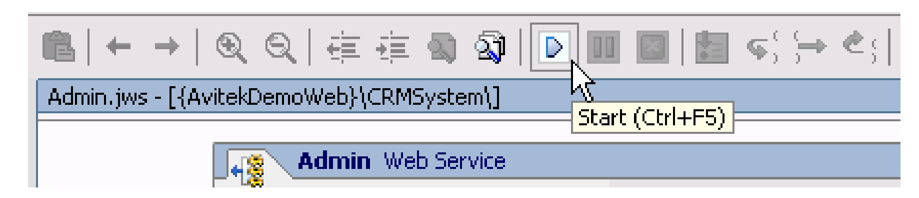


圖 8. 使用工具列來開啓 *Admin.jws* Web service 應用程式

- 假如你還沒完成相關步驟，你將會被提示去開啓 WebLogic Server 來完成相關動作。開啓 WebLogic Server 大約需要幾分鐘時間才會完成。(注意：不要關閉"Starting WebLogic Server"視窗，關閉這個動作會造成正在啓動時停止 server。你可以透過按下 **Hide** 按鈕來隱藏視窗。)

自動部署(Automated Deployment)

在建置完 *Admin.jws* 應用程式之後，它會自動被部署到 BEA WebLogic Server 8.1 上，而且 *Workshop Test Browser* 將自動顯示在畫面上。這證明了 WebLogic Workshop 自動部署和整合測試功能啓動了，你可以透過範例 3 來瞭解更多細節。

- 利用 *Admin* Web service 建立與載入樣本資料庫，在 Workshop Test Browser，啓動 *Setup* 功能透過按下 **Setup** 按鈕，如**圖 9** 所示。這個功能將建立與載入樣本的客戶和訂單資料庫欄位。*Setup* 功能提供連接這兩個資料庫的 Java 控制項，在接下來的章節，將可以學到更多有關 Java 控制項。
    - 注意：假設你是第二次執行這個範例，而想要清除舊的測試資料，請先選擇 *Reset* 功能，然後再執行 *Setup* 功能。

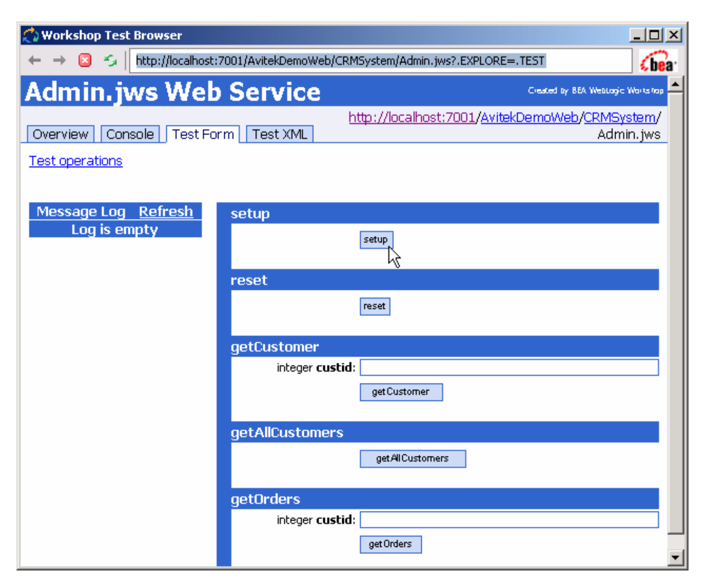


圖 9. Workshop Test Browser 顯示客戶端測試介面

- 在 *Setup* 功能被執行後，你可以透過讀取在 Workshop Test Browser 所描述的事件，來追蹤客戶與訂單資料庫的相關動作。你也可以按下 **Test Operations** 連接的測試客戶端來執行其他的 Web service 方法，如 ***getCustomer*** 或 ***getOrders***。
- 回到 BEA WebLogic Workshop Design View，關閉 *Admin.jws* 檔案(按下右上角的 X 按鈕)，接下來你可以開始範例 2，製作控制項。.

範例 #2. 建立一個客制化的控制項

Java 控制項 在 Workshop 程式設計模組裡是一個重要的中心元件，它可以使得開發者輕易地連結到現存的資料、系統、應用程式和商業邏輯。Java 控制項實質上是一個有方法、事件、屬性的商業邏輯元件，可以讓你視覺化地、宣告地指定它的行爲、集中在處理事件和呼叫方法，而不用撰寫複雜的 J2EE 運用大量 API 的程式碼。

內建的 **Java** 控制項

在 Java 控制項的實作方面，讓我們來探索預先定義在 Order Entry Demo 樣版裡面的 Avitek 資源的 Database 控制項。

- 在 Application Window 裡，找到 CRMSystem 資料夾，雙擊 CustomerDB.jcx 檔案。這將會載入 Avitek Customer Database 控制項到 Design View 視窗裡。

如圖 10 所示，CustomerDB 控制項就像所有的 Java 控制項一樣，可以透過在 Design View 視窗裡視覺化地表示來呈現給使用者，並且在視窗的左方列出它的方法和事件。.

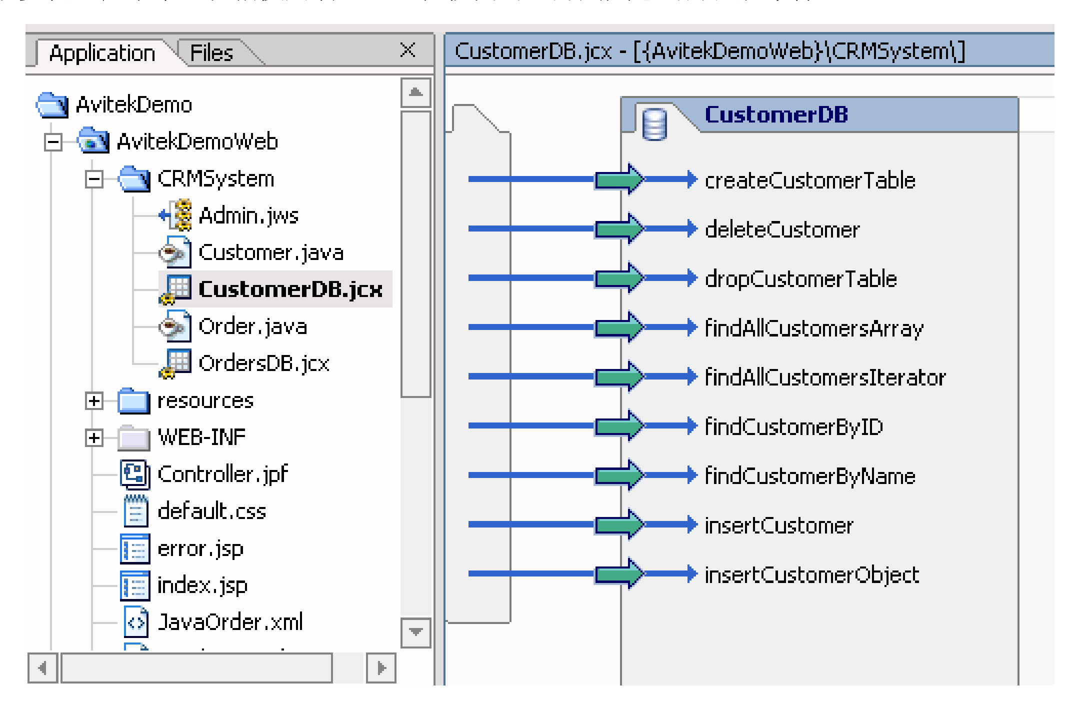


圖 10. CustomerDB Java 控制項在 Design View 的模樣

更進一步地瞭解資料庫控制項如何允許開發者輕易地運用任何的 SQL 資料庫，而不用知道 JDBC API 的命令，試著下列步驟：

- 雙擊 insertCustomerObject 方法的箭頭圖示。這將帶出一個對話盒(如**圖 11** 所示)，在這裡面開發者只要將 SQL 參數對應到適當的 Java 方法定義。這是一個 Java 控制項如何對開發者隱藏 J2EE 複雜性的例子。大多數的應用程式開發者知道 SQL，但是有很多開發者不知道(或想要知道)JDBC。當你探索 SQL/Interface Editor 後，點擊**取消**。

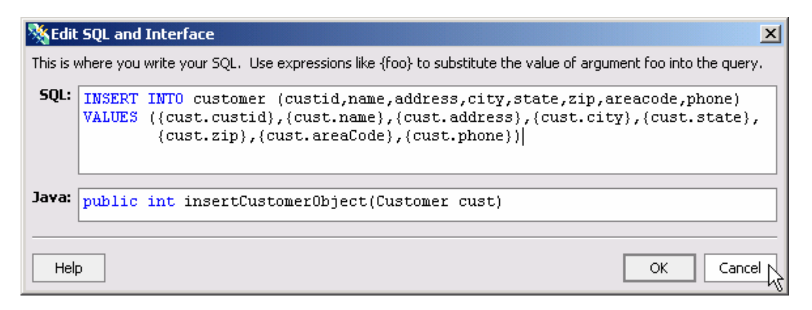


圖 11. 探索 SQL and Interface Editor 來對映 SQL 陳述式到 Java 方法

Database 控制項是 WebLogic Workshop 8.1 包含的標準 Java 控制項之一。其他內建的 Java 控制項提供對主要企業資源如 EJB component、Web services 與 JMS 佇列一個簡易存取的方式。這些控制項對於容易使用性與避免開發者需要寫複雜的、物件導向的 J2EE 架構程式碼來實現需要的整合目標是一致的。你可以點擊 Data Palette 上的 New 按鈕來看 Workshop 內建的控制項。

客製 Java 控制項

除了內建的 Java 控制項之外，WebLogic Workshop 8.1 還提供了一個可擴充的控制項架構，以致於開發者有能力來創造可重複使用的商業邏輯元件，並且可以與任何企業資源、ISV 應用程式或是一部份的商業邏輯作整合。客製 Java 控制項可以提升內建 Java 控制項的功能、自我套疊 (nestable)，並且可輕易地重複使用。此外，任何的 Java 控制項可以合併在任何或全部延伸至 Web services、Web applications 和其他 WebLogic Platform 應用程式的專案裡。

替 CRM System 創造客制化的 Java 控制項：

- 關掉在 Design View 裡的 CustomerDB 控制項。要開始建立一個新的控制項，在 Application Window 裡的 CRMSystem 專案目錄夾上按右鍵，選擇 **New→Custom Java Control**。

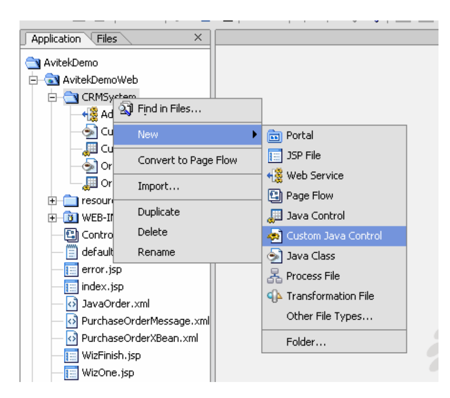


圖 12. 創造 CRMControl Custom Java Control

- 在 New Control 對話盒裡鍵入 CRMControlImpl.jcs 的名字並按下 **Create**(所有的 Custom Java 控制項必須要以"impl"結尾，雖然剩餘導讀的部分我們還是只會稱它為 CRMControl)。這將會導致一個新的客制化 Java 控制項被顯示在 Design View 裡。既然它並沒有任何的元件，它現在只顯示出一個空的視窗而已。

新客制化 Java 控制項的目的是提供一個單一的元件，依序處理協調 Avitek 有關於訂單管理程序的現存資料庫系統。尤其，任何你建立的 Order Entry system 將會需要與 Customer Database 和 Order Database 溝通。就像之前所提到的，你已經使用到可以便利這項整合的 Java 控制項。當這個控制項一被建立，它可以輕易地從任何其他的 WebLogic 平台應用程式再利用。

- 爲了開始建立 CRMControl，你必須先新增它所需的後端資源。你可以只不過從 Application Window 拖放 CustomerDB.jcx 項目到 Design View 視窗上的 CRMControl 圖示。

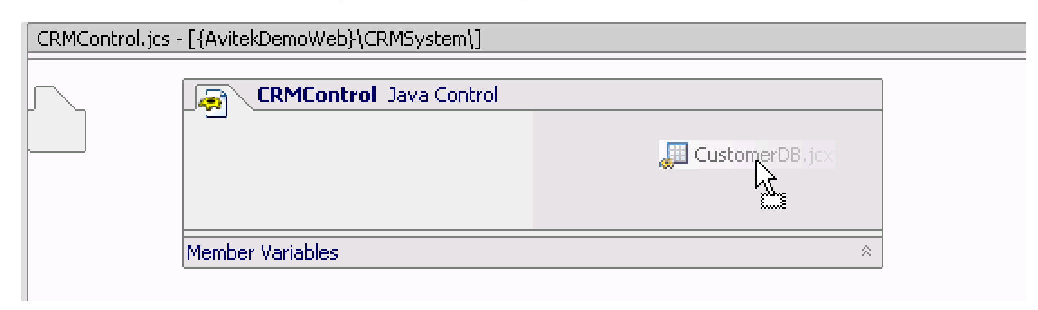


圖 13. 拖放 Customer Database control 到 CRMControl 視窗

- 同樣的，從 Application Window 拖放 OrdersDB.jcx 項目到 CRMControl 視窗。

就這兩個簡單的拖放動作，你已經完成了整合你的客制化 CRMControl 與 Avitek 資料庫的程序。甚至不用寫任何一行程式碼！ Design View 將會反映出這些新增的項目，如圖 14 所示。

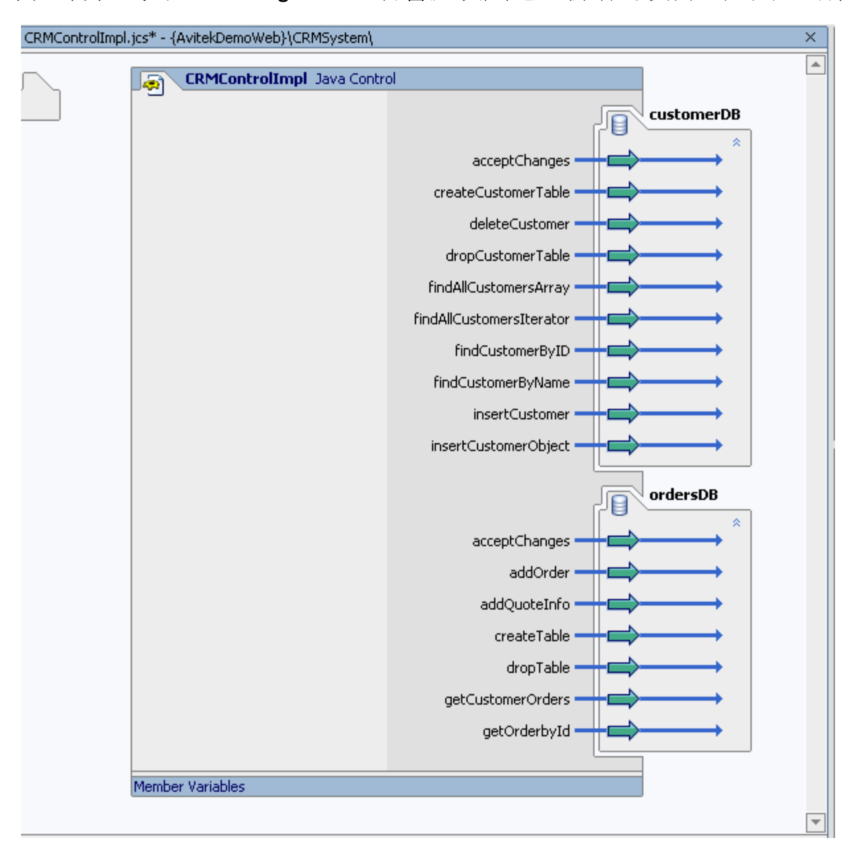


圖 14. Design View 反映出資料庫控制項新增到 CRMContro

下一步，你需要具體說明用戶端可以用來呼叫 CRMControl 的那個方法。Workshop 自動地移到 Data Palette 視窗，並將方法關連到已經新增到專案的 Java 控制項，使得這個步驟同樣地簡單。這些”傳過去(pass-through)”的方法可以馬上使用來提供用戶端存取任何控制項的方法。

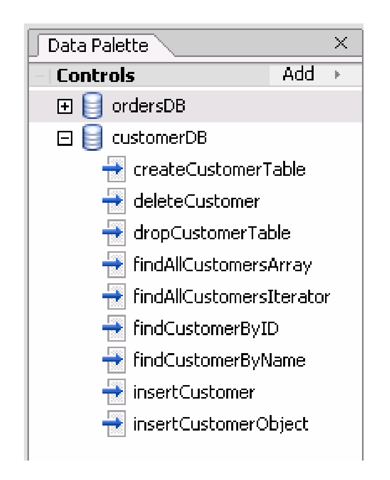


圖 15. 在 Data Palette 中的"pass-through"方法列表

➢ 為了提供 CRMControl 所想要的功能，你需要提供存取到 Customer Database 控制項的三個方法：findAllCustomersArray、findCustomerByID 與 insertCustomerObject。只要在 Data Palette 裡找到相對應的項目(如**圖 15** 所示)，然後再將每一項目從 Data Palette 拖到 Design View 裡的 CRMControl 上頭。這提供了一個快速方便的方式使得現存功能在控制項裡是可利用的。如果額外的商業邏輯需要的話，這些方法可以允許稍後做客制化的修改。

CRMControl 在 Design View 裡頭將會被更新，如以下**圖 16** 所示

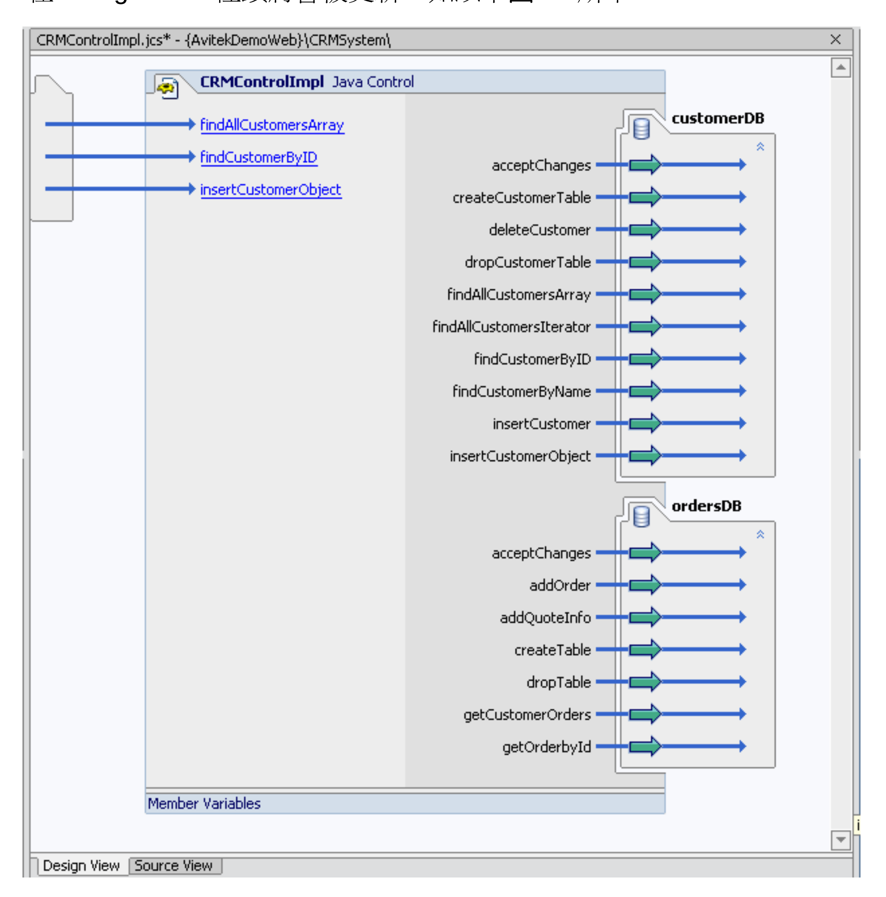


圖 16. 在新增"pass-through"方法之後 CRMControl 所看到的樣子

除了以上三個方法外，你需要新增一個新的方法來允許客制化 CRMControl 的客戶端來提交一張新的訂單，並且對 Customer 與 Order 資料庫且作適當的處理。

- 為了創造一個新的方法，將"Method"圖示從 Palette 視窗(左下角) 拖到 Design View 裡的 CRMControl 上頭。將這個新方法的名字定義為 createOrder。

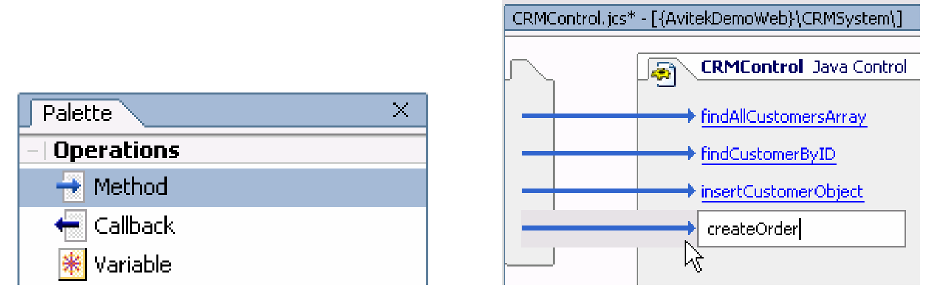


圖 17. 用方便的拖放方式來實作 createOrder 方法

createOrder 方法需要接受有關於一張訂單和其顧客的資訊。如果一份適當的紀錄不存在時，它還要開始在 Customer 資料庫裡建立一個新的顧客。最後還要新增相關的訂單明細到 Order 資料庫。實作這個商業邏輯並且與 Avitek 的資料庫作整合，將會需要一些程式碼：

- 從 Design View 裡，點擊 createOrder 的超鏈結來觀看在 Source View 裡頭的這個方法。



圖 18. 超鏈結使得從 Design View 轉換到 Source View 變得更簡單

- Source View 提供了一個方便的環境來撰寫需要的 Java 程式碼，並且直接寫到來源檔案。你需要加上下列 createOrder 方法的程式碼：

```
public String createOrder(Customer c, Order o)
{
   java.util.Random r = new java.util.Random();
   if (customerDB.findCustomerByID(c.custId) == null)
   {
         c.custId = r.nextInt(100000);
         customerDB.insertCustomer(c.custId, c.name,
            c.address, c.city, c.state, c.zip, c.areaCode,
            c.phone);
   }
         ordersDB.addOrder(c.custId, r.nextInt(100000),
            o.cameraQty, o.flatScreenQty);
         return "Thank you for your order, " + c.name +
". Order processed for " + o.flatScreenQty +
" Flat Screen TV sets and " + o.cameraQty +
" Digital cameras.";
}
```

當你完成鍵入程式碼，這個方法就像以下所示：

```
    /**
     * @common:operation
     */
public String createOrder(Customer c, Order o)
{
    java.util.Random r = new java.util.Random();
    if (customerDB.findCustomerByID(c.custId) == null)
    {
        c.custId = r.nextInt(100000);
        customerDB.insertCustomer(c.custId, c.name, c.address, c.city,
            c.state, c.zip, c.areaCode, c.phone);
    }
    ordersDB.addOrder(c.custId, r.nextInt(100000), o.cameraQty, o.flatScreenQty);
    return "Thank you for your order, " + c.name + ".  Order processed for " +
        o.flatScreenQty + " Flat Screen TV sets and " + o.cameraQty +
        " Digital cameras.";
}
```

圖 19. 在 Source View 裡看到的 createOrder 方法

注意，CRMControl 所需的功能可以很容易地用幾行 Java 程式碼寫出來，還有跟後端資料庫溝通就好像跟任何本端 Java 類別一樣。開發者可以從低階的 J2EE 複雜度萃取出來，以增加開發者的生產力。

非同步作業控制項

因為 Avitek 可能需要同時處理大量的訂單，你想要確認 CRMControl 不管負載有多重都可以可靠地回應這些需求。畢竟如果 Avitek 失去了這些訂單需求，他們也就等於失去了生意。在這個情境下的一個好的架構方法就是去實作一個同步化的解決方案。舉例來說， CRMControl 將接踵而來的訂單需求放置到一個可靠的佇列裡，那裡它保證直到 CRMControl 準備要要處理那個需求依然存在，而不是一外部的客戶端送出一訂單需求並且只是無限期的等待訂單處理。

支援同步化的溝通是一個建立大規模的、可靠的企業級應用程式重要的元件。然而由於複雜的程式設計需求與對 J2EE 科技的良好認知，如 Message-Driven Beans (MDB)和 JMS 佇列，使得它變成一個非常困難的任務。好在 BEA WebLogic Workshop 8.1 讓實作同步化就像設定屬性一樣簡單。為了展示，你將需要創造一個新的方法來提供全部的客戶端存取到那個客制化的 CRMControl。

- 從 Design View 將"Method"圖示從 Palette 視窗拖到 Design View 裡的 CRMControl 上頭以插入一個新的方法。同時定義這個新方法的名字為 createOrderAsynch。
- 為了要讓 createOrderAsynch 這個方法 active(如果還沒 active 的話，點擊這個方法的箭頭圖示)，存取 Property Editor 然後再設定 Message-Buffer Enable 屬性為 **True**，如**圖 20** 所示。

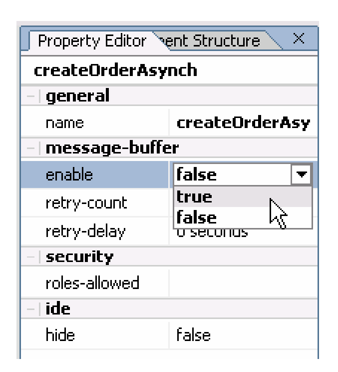


圖 20. 使用 Property Editor 來開啓 message-buffering

你將會注意到 Design View 會自動更新為一個佇列的圖示在 createOrderAsynch 之前來指明這個方法將會在同步化的模式下與其他的方法溝通。

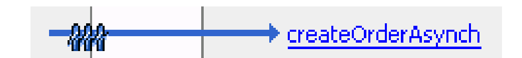


圖 21. Design View 新增一個佇列圖示來代表非同步模式

結果 createOrderAsynch 方法不能像 createOrder 方法一樣只是回應到呼叫的那個客戶端。取而代之的是，你需要創造一個 Callback 來允許 CRMControl 在 createOrderAsynch 已經完成處理那張提交的訂單之後通知客戶端。

➢ 爲了創造一個新的回傳，將"Callback"圖示從 Palette 視窗拖到 Design View 裡的 CRMControl 上頭。定義這個新回傳的名字爲 orderResponse。

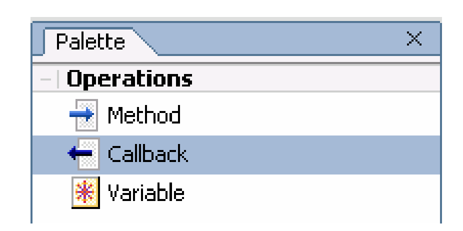


圖 22. 從 Palette Window 拖放適當的圖示來創造一個新的 Callback

就像你之前 createOrder 方法做的一樣，你將需要提供商業邏輯給 createOrderAsynch 方法。

➢ 點擊 createOrderAsynch 上的超鏈結轉換到 Source View。

➢ 幸運的是 createOrderAsynch 所需商業邏輯的對應程式碼已經提供給 createOrder，所以你只需要修改以下兩行程式碼：

```
public void createOrderAsynch(Customer c, Order o)
{
    callback.orderResponse(createOrder(c,o));
}
```

在作了這些修改之後，你可能會注意到最後一行的底下有劃紅線，指明了 source code 裡有一個錯誤，如**圖 23** 所示。

```
/**
 * @common:operation
 * @common:message-buffer enable="true"
 */
public void createOrderAsynch(Customer c, Order o)
{
    callback.orderResponse(createOrder(c,o));
}
```

圖 23. 自動語法檢查對原始碼錯誤提供即時的警告

Workshop 在程式碼行間的語法檢查與通知使得開發者更容易修正錯誤，當他們在同一內文裡寫程式時就能修正錯誤，而不用等到編譯時才發現。這個錯誤的理由是 orderResponse 回傳的呼叫並沒有對應到當你在 Design View 裡新增 orderResponse 操作時 Workshop 自動創造的那個宣告。

- 爲了要作這簡單的修改，回到 Design View 上然後再點擊 orderResponse 回傳就能看到原始碼。到 orderResponse 宣告的 Source View，然後再新增一個 String 型態 message 參數。當你完成以後，宣告看起來像是這樣：

```
{
    void orderResponse(String message);
}
```

- 轉換回 Design View 來看完成的 CRMControl。

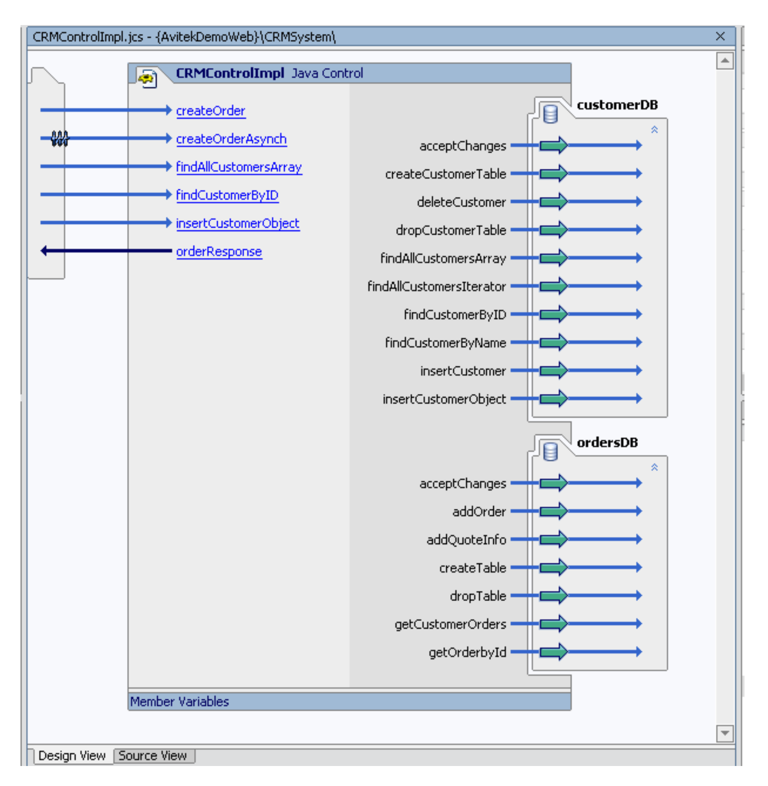


圖 24. 在 Design View 中已完成的 CRMControl

在以上的幾個步驟裡，你已經創造了一個分散式的、可重複使用的商業邏輯元件，用來提供同步與非同步的存取到 Avitek 的訂單管理資料庫。在以下的範例中，你將會建立一個 Web service 和一個 Web application，這兩個都會利用到你剛剛創造的 CRMControl。

範例 #3. 新建企業級的網路服務

你在範例 2 中實做的客製 CRMControl 接收了顧客及訂單的資訊，並做了所有必要的後端處理，以及對 Avitek 的系統來說必要的資料庫同步 (database coordination)。然而，你還沒有替外部使用者提供 access CRMControl 介面的機制。在這部分的練習中，你將會透過一個馬上允許數千個企業伙伴用 Avitek 的 Order Entry 系統進行跨組織操作的企業級 Web service 應用系統來接曩 CRMControl 的功能。

- ➢ 首先，儲存及關閉所有開啓的檔案以清空工作區.
- ➢ 爲了產生一個新的 Web service，請用滑鼠右鍵點 Application 視窗中的 *AvitekDemoWeb* 專案圖示，選擇 **New → Web Service**，再將這個 web service 命名爲 *orderEntryService.jws*.

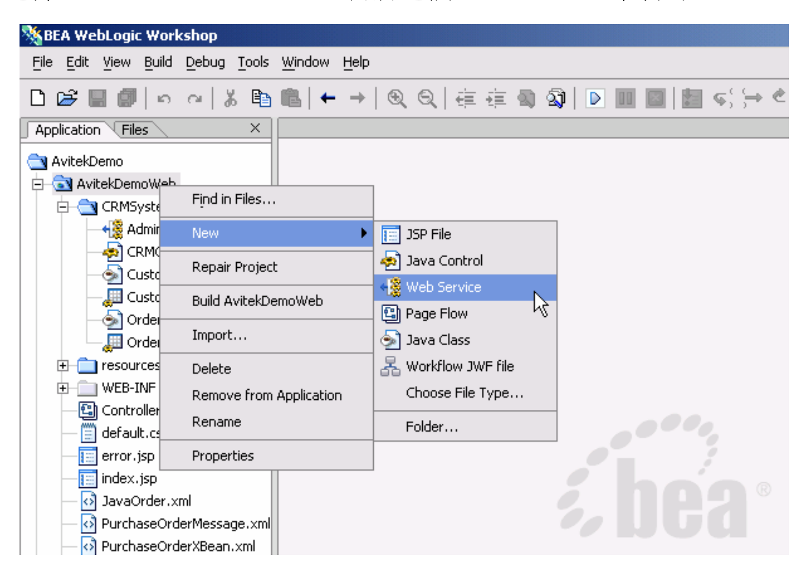


圖 25. 產生一個名爲 *orderEntryService.jws* 的新 Web service.

正如你可能預期的，你現在將會在 Design View 中看到一個 *orderEntryService* 應用系統的圖形化描述，而目前它只是一個空的框框。一般而言，如果你想在 Web service 應用系統中包含任何內建的 Java 控制項，你可以從 Control Palette 中選擇適當的控制選項即可。拜你在之前範例的努力之賜，你只需要在這個 *orderEntryService* Web service 中運用客製的 CRMControl.

- ➢ 爲了運用 CRMControl，請從 Application 視窗中選擇它的圖示，然後把它拖曳到 Design View 中的 *orderEntryService* 框框下。



圖 26. 欲增加到 CRMControl 的 access，只要將它的圖示拖曳到 orderEntryService.jws 框框下。

現在 *CRMControl* 出現在 Design View 框框的右側，表示它對開發者而言是一個容易運用的可用資源。

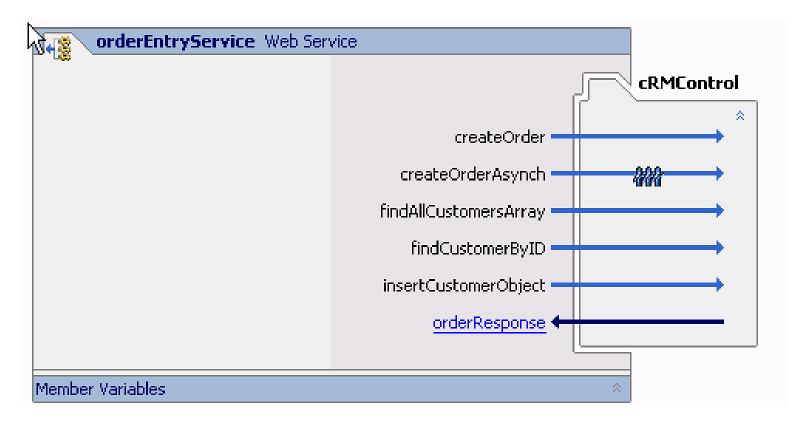


圖 27. Design View 顯示 *CRMControl* 為在 *orderEntryService.jws* 中的一個後端資源

接下來，你需要建立可以被這個 Web service 應用系統的客戶端呼叫的方法。再一次地，你會很容易地做到這點，因為 BEA WebLogic Workshop 會自動產生 "穿透" 方法來存取 CRMControl.

- 欲增加想要的方法，只要將 *createOrderAsynch* 實體和 *orderResponse* 實體從 Data Palette 中拖曳 *orderEntryService* 框框下。這些穿透的方法允許這種透過 Web service 應用系統的方式，便利地呈現 CRMControl 的方法 。

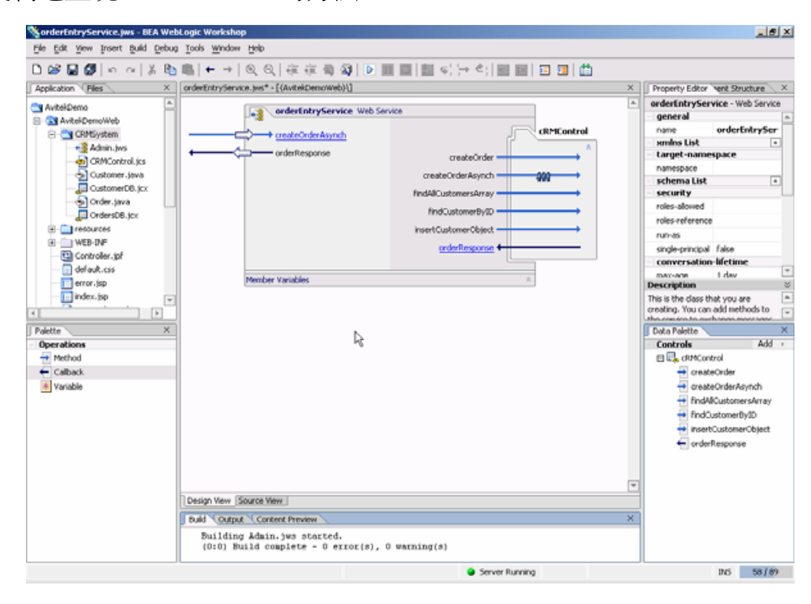


圖 28. *createOrderAsynch* 和 *orderResponse* 的操作被呈現給客戶端。

有了面對客戶端的方法和適當的後端資源，現在是一個好機會來探究 WebLogic Workshop 8.1 中，用來幫助建立企業級 Web services 的額外特色了。例如：非同步、鬆散耦合、堅固的訊息層安全以及可靠的 "一次且僅一次 (once and only once)" 訊息；這些對於建立起可靠及具有擴充性、幫助顧客解決真實商業問題的 Web services 來說，都是非常重要的特色。並不令人意外地，Avitek 會想要利用這些進階建構的特色。

支援非同步 Web Services

在真實世界中，大部分的 Web services 應用系統並不能立即傳回一個答覆。它們可能需要存取一個既有的系統、聯絡第三方，或者牽涉到一個人類使用者。這些操作在本質上就是非同步的，所以

讓非同步的 Web services 支援盡可能簡單是很重要的。有了 BEA WebLogic Workshop 8.1，讓非同步的 orderEntryService 簡單到設定幾個屬性而已。

- 在 Design View 中啓動 *createOrderAsynch* method (如果必要的話，用滑鼠點在 method 箭頭圖示啓動它)，點選 Property Editor 可以看到各種客製化設定都是可用的。將 *Message Buffer Enable* 屬性設爲 **True**，這會產生一個佇列圖示被加到此 方法的視覺化圖形畫面。

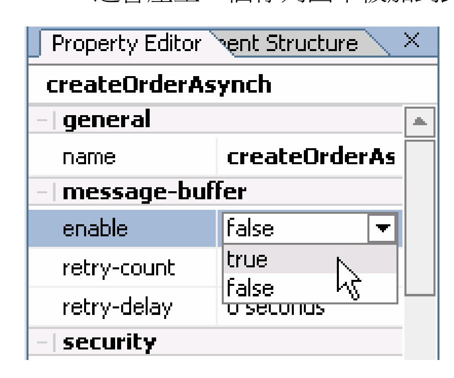


圖 29. Use the Property Editor to declaratively specify complex application functionality

正如你已在前面範例中所見的，啓用訊息緩衝器 (message-buffer) 屬性會導致一個 JMS 佇列和相關的訊息基礎建設在執行的時候被實作。在這個案子中要注意的是，這個佇列是被建立在 *orderEntryService* 應用系統和它的客戶端之間；而在前面的範例中，*CRMControl* 是在 *CRMControl* 和它的客戶端 — 在這個案子中就是 *orderEntryService* 應用系統， 之間實作一個非同步的 method 來互動。因此，非同步的兩個層次對於最佳的應用系統擴充性及效率是非常有效地。

- 在 *createOrderAsynch* method 仍在啓動狀態下，將 *Conversation* 區域 設置在 Property Editor，然後將這一階段設爲 **Start**。這項設定會反應出一個綠色的三角形加在此 方法的視覺化圖形畫面。
- 最後，用滑鼠點在 Design View 中的 *orderResponse* callback，然後將它的交談階段 (Conversation phase) 改爲 **Finish**。這項設定會反應出一個紅色的正方形加在此 callback 的視覺化圖形畫面。

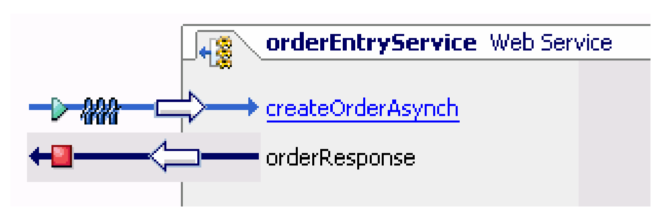


圖 30. 利用 WebLogic Workshop 的交談設定來管理非同步通訊。

這些交談設定是基於 Workshop 對非同步通訊的交談暗喻，這些設定幫助應用系統持續追蹤來來回回地訊息流動，以及確保所有同一談話之一的訊息都送到對的地方 (例如：打電話中的客戶)。甚至，交談會用一種堅強可靠的方式 (利用實體 EJBs)，自動地儲存任何有關此談話的狀態 — 開發者只需要在 JWS 類別中宣告變數來利用這項功能。藉由利用宣告屬性設定的方式建立這些 methods 的交談本質，BEA WebLogic Workshop 8.1 自動處理了狀態管理 (state management) 的細節、訊息關聯 (message correlation) 和其他更多。

BEA WebLogic Workshop™ Property Editor 是非常有意義的，因爲它不只是讓開發者以視覺化的方式更改簡單的值。一些屬性像是交談階段和訊息緩衝等，接棄了一般來說可能需要寫 40 至 80 行 JAVA 程式碼以及了解 JMS 和 JNDI APIs 的複雜功能。屬性讓這些強大的 J2EE 概念對所有開

發者來說都是可行的，而不需有 J2EE 專家知識的層次。爲了更加了解 Workshop 如何這麼簡單地實現其強大的功能，你可能需要花幾分鐘來理解屬性設定是如何被記錄的：

➢ 從 Design View 中，用滑鼠點在 *CreateOrderAsynch* 的超連結上以切換到 Source View.

```
/**
 * @common:operation
 * @common:message-buffer enable="true"
 * @jws:conversation phase="start"
 */
public void createOrderAsynch(CRMSystem.Customer c, CRMSystem.Order o)
{
    cRMControl.createOrderAsynch(c,o);
}
```

圖 31. Source View 對兩種編輯方式提供一字不漏地同步

請注意原始碼的註解是位於方法的宣告之上，在 JavaDoc 註解之中。這些程式碼註解是在佈署期間時被 Workshop 執行期框架所解譯，然後 Workshop 會自動在產生的應用系統中啓動這個功能。這項非凡的能力，伴隨著以事件爲基礎的 (event-based) 視覺化設計環境，就是對所有的開發者 — 不僅僅是 J2EE 的專家們 — 而言，在 WebLogic Enterprise Platform 上有成果地工作是這麼簡單的原因。

需要注意的是 WebLogic Workshop 並不會產生程式碼 — 開發者永遠不需要去了解或除錯那些他們不曾寫過的程式碼。取而代之的是，屬性和它們產生的註解會宣告在伺服器中被執行期框架所支援的功能。

僅僅在這幾個步驟當中，你此刻已產生了一個 Avitek 可以馬上佈署在批發商區域以在既有的網際網路基礎建設上自動定購的強大 Web service 應用系統。請注意創新的 WebLogic Workshop 程式設計模型和 run-time 架構，讓你從繁瑣的 SOAP 參數解讀、Bean 的佈署、WSDL 檔案的產生，以及更多的細節中抽象出來。用這種方法，開發者不再受限，而可以專注於處理事件和撰寫商業邏輯，讓它們有更戲劇性地大的生產性。

自動化部署與測試

你現在已準備要試行 *orderEntryService* 了：

➢ 爲了自動地佈署及執行這個 web service，請用選單選擇 **Debug → Start**，或用滑鼠點擊選單列上的 **Start** 圖示。

當你啓動 *orderEntryService* 應用系統， BEA WebLogic Workshop™ 8.1 便會在 WebLogic Server 上自動編議及佈署 Web service 應用系統。請注意你不需要擔心撰寫佈署描述子、設定應用系統伺服器，或打包你的應用系統元件。取而代之的是，BEA WebLogic Workshop "撰寫且執行(Write and Run)" 特色能讓開發者快速地建立和佈署應用系統，戲劇性地減少反覆開發過程中所需要的時間。

由於 Web services 典型地會牽涉到應用系統對應用系統的通訊，所以在開發的時候測試 Web services 一直以來都是冗長乏味以及花時間的。事實上，你通常必須撰寫客戶端的程式碼來測試 web service。然而爲了簡化測試，BEA WebLogic Workshop 提供了一個以瀏覽器爲基礎 的介面來測試所有的應用系統，如此一來便可以驗證功能以及快速地確認問題。這項整合的測試安全措施幫助 WebLogic Workshop 的模型做到快速的應用系統開發 (rapid application development) 。

➢ 從 Workshop Test Browser 之中，用滑鼠點 **Overview** 頁籤。



圖 32. WebLogic Workshop 8.1 generates 可產生一個測試客戶端 (test client) 來快速地測試任何應用程式。

Workshop Test Browser 的預覽畫面 (Overview) 會自動提供 Web service 及其所呈現的方法的一個基本描述。它允許客戶端輕易地下載一個 WDSL 檔、一個 Java 控制項 可以存取的 *orderEntryService* 應用系統原始碼，或者用在任何 Java 環境以替這個 Web sevice 建立一個客戶端的 Java 代理執行程式。同樣地，用滑鼠點在 Console 頁籤將能讓開發者看到這個 Web service 正在存取什麼樣的服務和元件。

➢ 為了測試 *orderEntryService* 應用系統，請選擇 **Test XML** 頁籤。

Workshop Test Browser 提供了一個預先載入 XML 訊息架構的開放文字區域，而這個 XML 訊息架構即為 orderEntryService 所期待進來的訂單資訊。

➢ 你可以人工地將值敲進 Workshop Test Browser，或甚至更方便地，貼在 *JavaOrder.xml* 檔提供給你的樣本訂購資訊。你可以將這個檔案置於 Application 視窗專案樹之下，用滑鼠點 *createOrderAsynch* 按鈕。

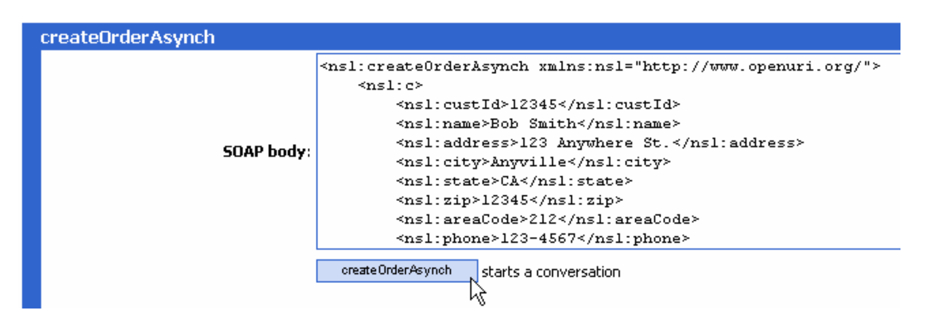


圖 33. 送出一個名叫 Bob Smith 的顧客的樣本訂單。

測試客戶端會自動將訂單資訊包入一個顯示在測試客戶端的適當 SOAP 訊息。嵌在 XML 中的為樣本顧客和訂單資訊：

Service Request createOrderAsynch
Submitted at Sun Feb 23 20:03:15 PST 2003

```
<SOAP-ENV:Envelope xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/" xmlns:SOAP-
  ENV="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <SOAP-ENV:Header>
    <StartHeader xmlns="http://www.openuri.org/2002/04/soap/conversation/">
      <conversationID>1046059394904</conversationID>
      <callbackLocation>http://TESTUI</callbackLocation>
    </StartHeader>
  </SOAP-ENV:Header>
  <SOAP-ENV:Body>
    <ns1:createOrderAsynch xmlns:ns1="http://www.openuri.org/">
      <ns1:c>
        <ns1:custId>12345</ns1:custId>
        <ns1:name>Bob Smith</ns1:name>
        <ns1:address>123 Anywhere St.</ns1:address>
        <ns1:city>Anyville</ns1:city>
        <ns1:state>CA</ns1:state>
        <ns1:zip>12345</ns1:zip>
        <ns1:areaCode>212</ns1:areaCode>
        <ns1:phone>123-4567</ns1:phone>
      </ns1:c>
      <ns1:o>
        <ns1:orderID>558488</ns1:orderID>
        <ns1:custID>12345</ns1:custID>
        <ns1:cameraQty>15</ns1:cameraQty>
        <ns1:flatScreenQty>8</ns1:flatScreenQty>
      </ns1:o>
    </ns1:createOrderAsynch>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Service Response
Submitted at Sun Feb 23 20:03:16 PST 2003

圖 34. Workshop Test Browser 協助追蹤所有被用在 Web service 的訊息。

回想一下你所測試的*createOrderAsynch* 方法是非同步的，而就像我們在設計街階段 (design-time) 所指出的，它開啓了一個交談 (conversation)。並不令人意外地，Message Log 顯示一個新的交談已被自動地啓動了。BEA WebLogic Workshop 會自動追蹤所偶在相同 Conversation ID的相關訊息，你可以在你的 Message Log 以及被送出的 SOAP 訊息檔頭 (headers) 看到。

- 用滑鼠點擊 Workshop Test Browser 的 **Refresh** 連結來瀏覽那些在這個非同步的交談中，已經在測試客戶端和你的 Web service 之間被交換的任何一連串訊息。很快地，你會看到從 *orderResponse* 客戶端傳回 (callback) 的訂單資訊。如 **圖 35** 所示：



圖 35. 非同步訊息的關聯是根據 Conversation ID.

- 在測試客戶端承認確認此訂單之後，請關閉 Workshop Test Browser 並且回到 WebLogic Workshop 的視覺化開發環境。

鬆散耦合(Loose-Coupling)

對於企業級的 Web service 來說，BEA 架構的另一項重要特色爲鬆散耦合 (loose-coupling)。鬆散連結的應用系統有彈性，且能隨著時間變遷而做出相應的改變，當 Web service 被用來整合那些通常由不同的團隊用不同的科技、不同的平台所建立的不同的應用系統，這一點就非常重要。鬆散連結的關鍵在於做到 Web service 的公用契約 (public contract) 和 它內部實作 (private implementation) 之間的分隔。既然 Web services 是用以 XML 爲基礎 (XML-based) 的文件來溝通；WebLogic Workshop 應用系統開發又是以 Java 來做的，那麼在 XML 和 Java 之間提供容易、聰明且彈性的對應 (mapping) ，以確保真正的鬆散連結是非常重要的。

在鬆散連結的幫助下，BEA WebLogic Workshop 8.1 對於 XML Schema 和 XQuery、在 Java 中使用 XML 等這些發展中的科技目前的進步水準都提供了深厚的支援。在接下來的數個步驟中，你將會把被商業伙伴們用來送出自動訂單給 Avitek 的 Purchase Order 訊息中，所提供的結構與限制 Schema 定義給具體化。你也將獲悉 XQuery 技術 — 其有時亦被稱爲 “針對 XML 的結構化查詢語言 (SQL)”，因爲它提供了堅實的機制直接從 XML 文件存取資料，就好像 SQL 提供在傳統資料庫中存取資料的機制一樣。幸運的是，BEA WebLogic Workshop 替你處理了撰寫顧客 XQuery 表達式的複雜工作。

- 你首先需要匯入 (import) 適當的 Schema 檔到 *AvitekDemo* 應用系統。在 Applications 視窗中，用滑鼠右鍵點擊 Schemas 資料夾，然後選擇 **Import**。瀏覽你之前存在 *c:\workshop81demo* 資料夾的 *purchase-order.xsd* 檔，然後點 **Import**。

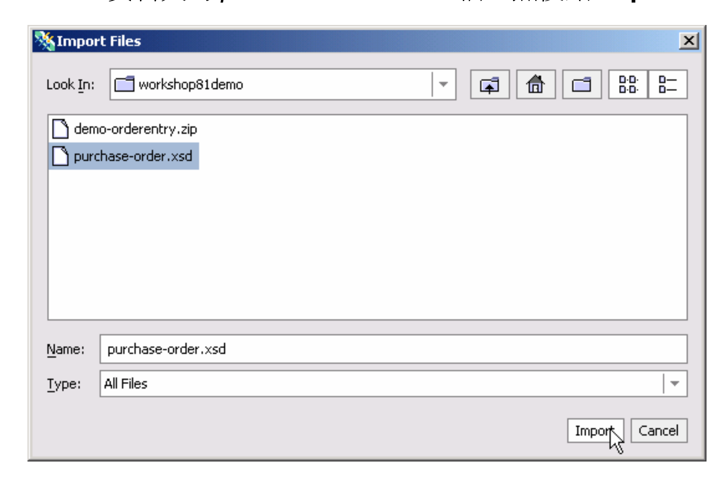


圖 36. 匯入 XML Schema Definition (XSD) 檔以用來做 XQuery 對應 (mapping)

BEA WebLogic Workshop 會自動地編譯 (compile) 和解譯 (interpret) 已提供的 Schema 檔，以學習 *orderEntryService* 應用系統應有的訊息結構。(你可以在 Design View 下面的 Build 面版看到這個編輯結果。)

接著，你將會需要提供在接踵而來的 XML 訊息中，資料檔的 “對應 (mapping)” 給替應用系統接取資料以便處理的 Java 資料結構中的元素。

- 從 *orderEntryService.jws* 的 Design View 中，用滑鼠點擊 *createOrderAsynch* method 的箭頭圖示來啓動這個 method。
- 在 Property Editor 中將下拉捲軸往下捲動，直到你看見在 *Parameter-XML* 這個標題之下的 *XQuery* 屬性爲止。用滑鼠點擊 **...** 鈕來啓動 *XQuery Editor*.

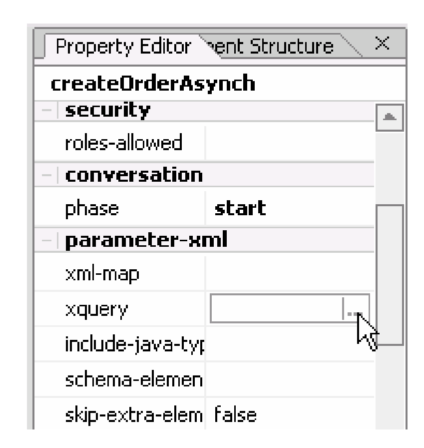
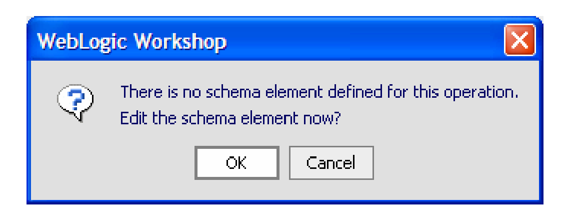
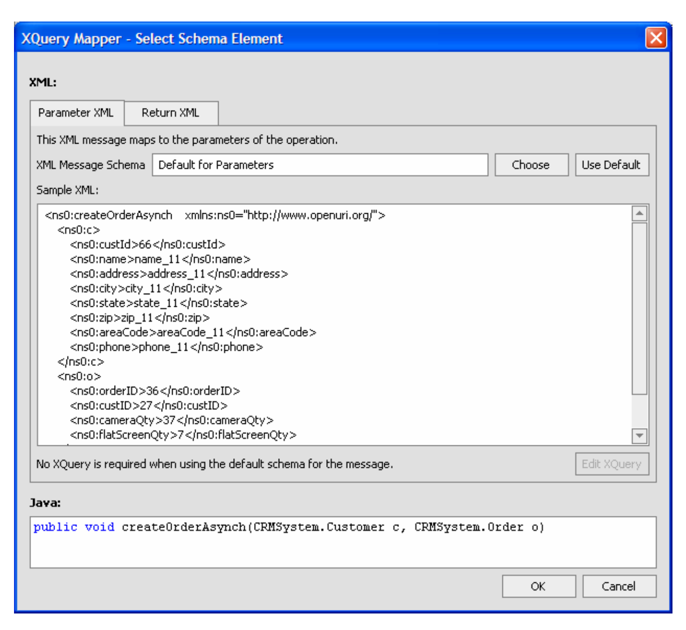


圖 37. 利用 Property Editor 來呼叫 XQuery Mapping 特色。

➢ 接下來，Workshop 會通知你尚未有 Schema 被聯結到 ***createOrderAsynch*** 操作上，你需要具體指定一個對應，因此請選擇 OK.



➢ Workshop 會呈現出目前的 Schema，預設 Schema 是根據 ***createOrderAsynch*** method 的 Java 特徵 (Java signature) 而定的。請選擇 **Choose** 按鈕以選擇一個新的 Schema.



➢ 在 *XQuery Editor* 中，如**圖 38** 所示，請展開樹狀結構以放置 *createOrderAsynch* Schema 項目。你需要將這個定義具體指定爲“來源 Schema”，所以請用滑鼠反白它然後點 **Create**.

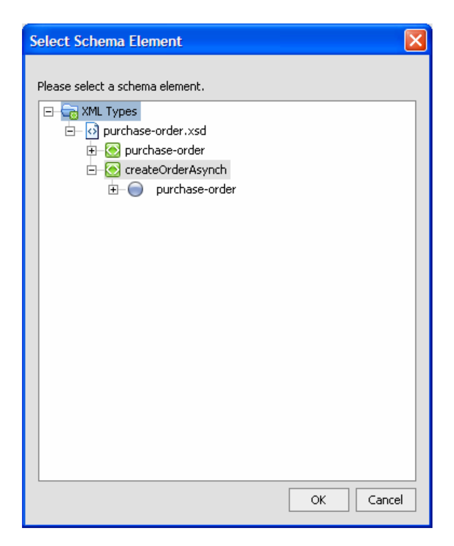


圖 38. 將 createOrderAsynch 定義具體指定爲來源 Schema

- Workshop 會出現另一個對話框，提醒你來源 Schema 已經和預設的不同了，你需要利用 Workshop 視覺化的 XQuery 編輯工具來做一些對應。請選擇 ***Edit XQuery*** 以從 **圖 38a** 的視窗中拉出視覺化的對應工具。

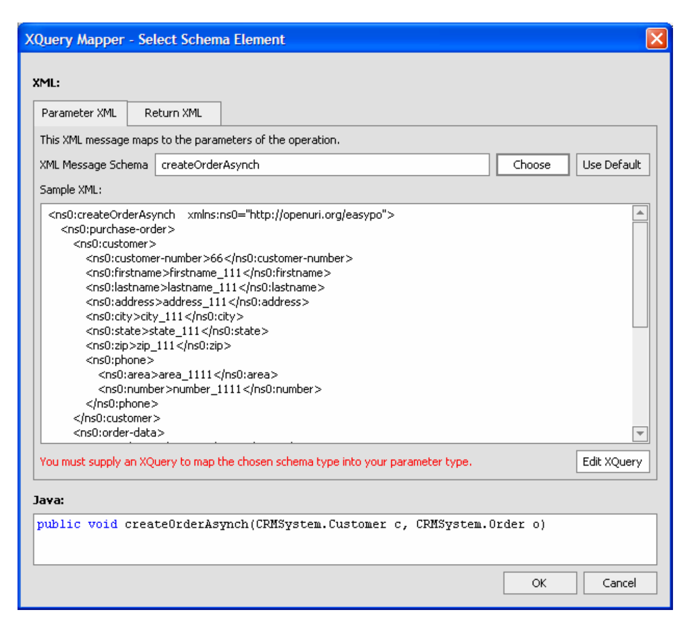


圖 38a. 選擇 ***Edit XQuery*** 來將新的來源對應到預設的目標。

接下來會出現一個視覺化的對應工具，將來自來源 Schema 的欄位 (基於已選定的 Schema 定義) 視覺化地對應到目的 Schema (一些被 createOrderAsynch method 宣告定義的參數，屬於 Customer 和 Order 資料結構)。在這兩個 Schema 之間的對應欄位，只是簡單地拉一條線將來將來源欄位連到目的欄位：

- 將 *customer-number* 連結到 *c.custId* 和 *o.custID*
- 將 *firstname* 連結到 *c.name*，然後將 *lastname* 同樣也連到 *c.name*
- 將 *address*, *city*, *state*, *zip*, 以及 *phone* 等欄位連結到 c 資料結構的相對欄位。

- 將 *order-number* 連結到 *o.orderID*
- 將 *tv-order:quantity* 連結到 *o.flatScreenQty*
- 將 *camera-order:quantity* 連結到 *cameraQty*

當你完成了這些對應以後，XQuery Mapper 的螢幕畫面應該會像 **圖 39** 所顯示的一樣。

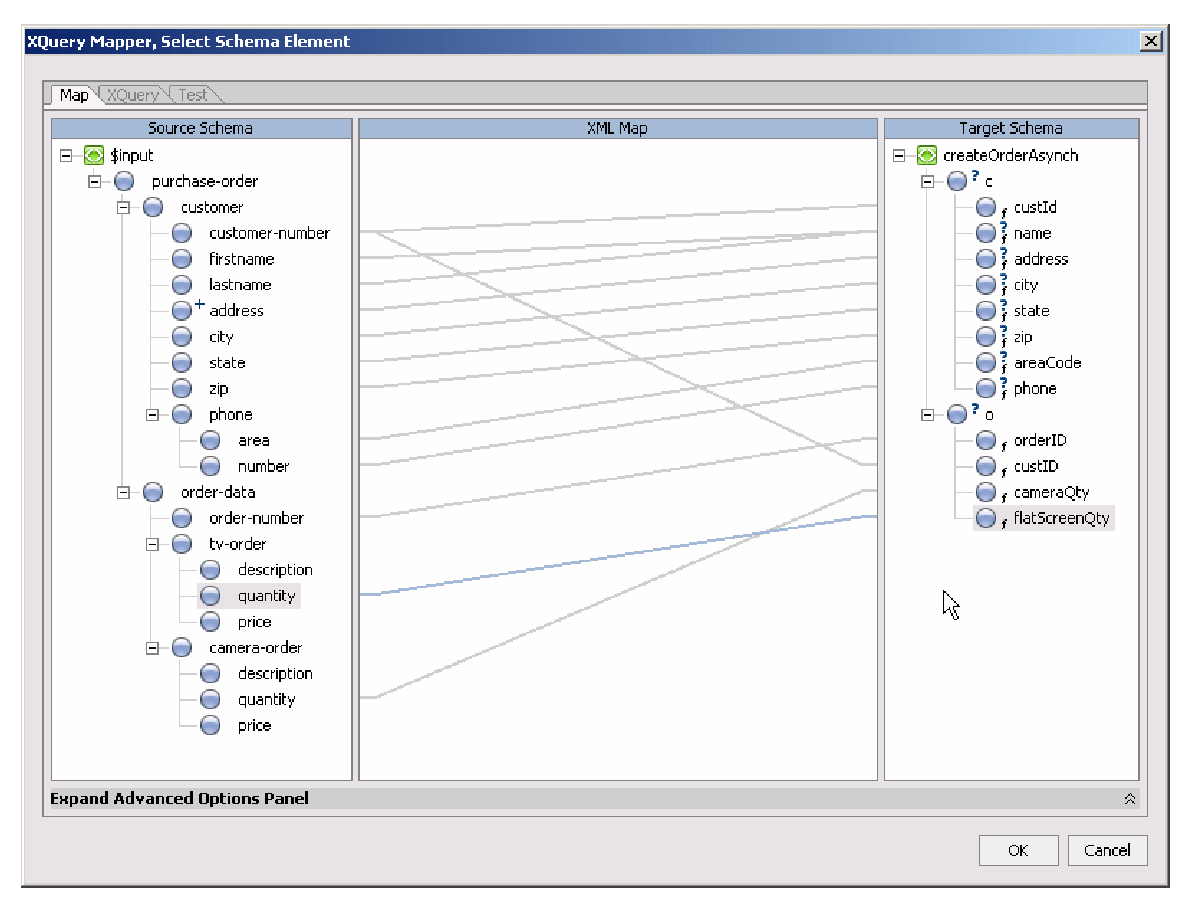


圖 39. WebLogic Workshop 實現了 Source (XML) 與 Target (Java) 之間的視覺化對應。

如上所示，你可以使用視覺化工具來加速，或者藉由點擊 XQuery 頁籤來直接讀取 XQuery 原始碼。每一個視覺化對應的動作都可以被轉換到相應的 XQuery 表達式。

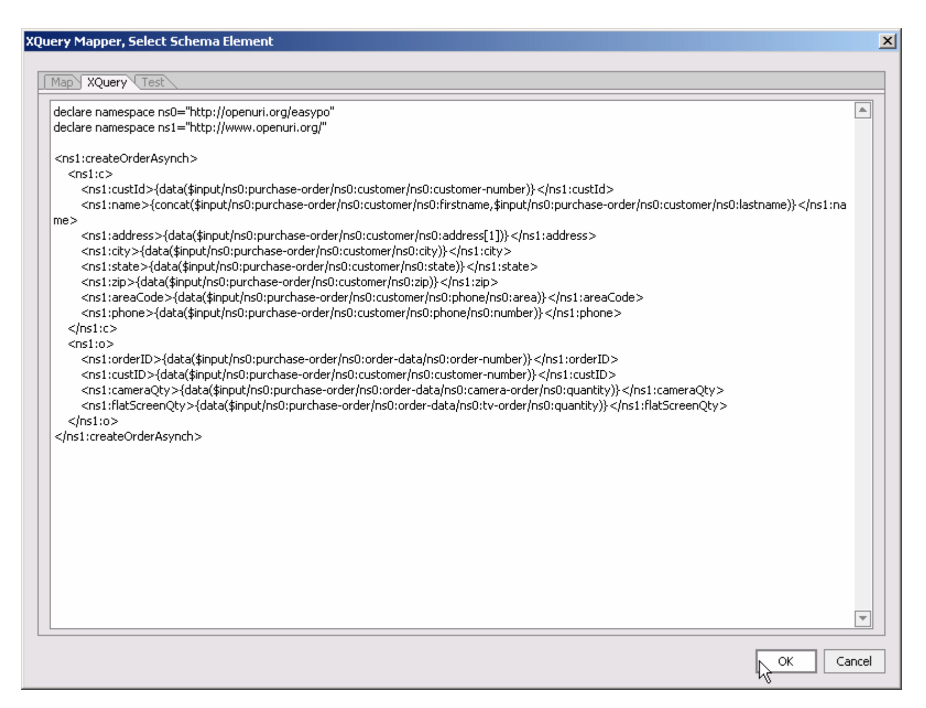


圖 40. Workshop 提供在對應之下的 XQuery 表達式程式碼檢視

- 請選擇在 *XQuery Mapper* 視窗上面的 *XQuery* 頁籤來檢視在原始 XQuery 格式之下的對應。爲了美觀完整起見，請在名字 (first name) 與姓氏 (last name) 之間，設定一個空白間隔 (“ ”) 來插入一些空間到名稱對應中。調整過的一行文字應該如下所示。

```
<ns1:name>{concat($input/ns0:purchase-
order/ns0:customer/ns0:firstname," ",$input/ns0:purchase-
order/ns0:customer/ns0:lastname)}</ns1:name>
```

- 請用滑鼠點擊 *XQuery Mapper* 工具上的 **OK** 鈕，然後在按 *XQuery Editor* 上的 **OK** 鈕以回到 Design View.

有了這些確保鬆散連結的 Web service 額外步驟，你現在就能測試你修改過的 *orderEntryService* Web service 應用系統了：

- 一開始，請利用選單來選擇 **Debug → Start**，或者點擊在選單列上的 **Start** 圖示。
- 請用滑鼠點擊 **Test XML** 頁籤。注意這次提供的 XML 文件外框 (XML skeleton) 和上次有些微不同，這是因爲 *orderEntryService* 應用系統現在正在等待一個符合你所指定之 *createOrderAsynch* Schema 定義的 XML 訊息。
- 爲了方便起見，一個符合預期 Schema 的 XML 訊息已被提供在 *PurchaseOrderMessage.xml* 檔裡頭了，可在 Application Window 中找到。請將這個 XML 訊息的全部內容都貼到測試客戶端的 SOAP 主體欄位中，然後再點擊 *CreateOrderAsynch*.

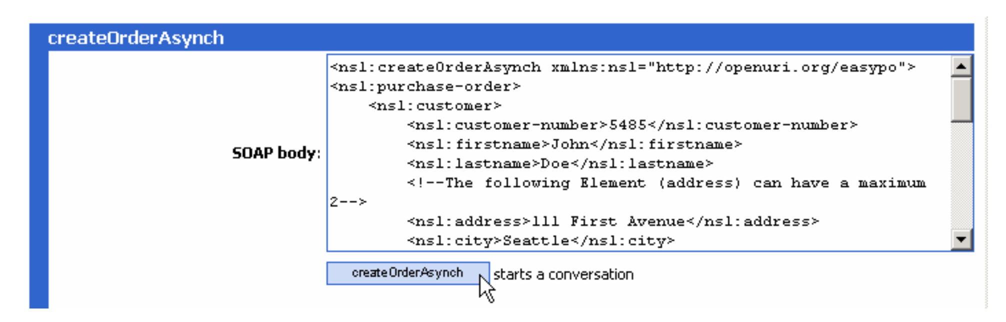


圖 41. 替一個名叫 John Doe 的顧客提交一份樣本訂單。

➢ 在呼叫這個非同步的 method 之後，你需要點擊在 Workshop Test Browser 上頭的 **Refresh** 連結來收集交談 (conversation) 中接下來的訊息。很快地，你會在確認此訂單的 Message Log 上收到客戶端的回應 (callback) 。

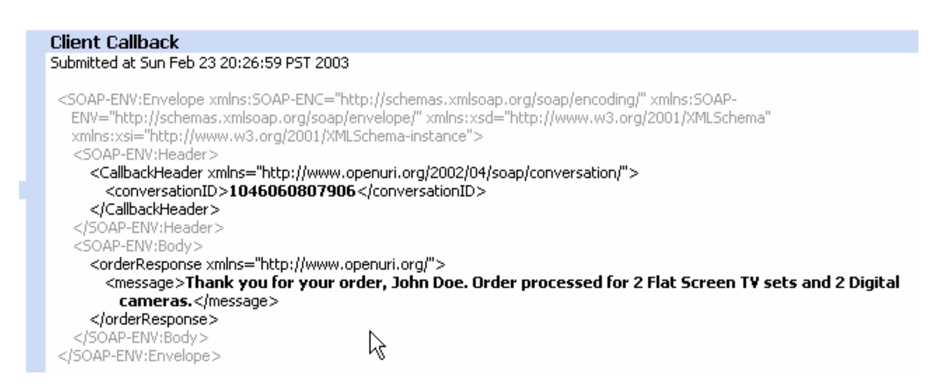


圖 42. John Doe 的訂單被 *orderEntryService* 應用系統確認收到了。

恭喜你！你已在幾分鐘內利用 **BEA WebLogic Workshop 8.1** 建置、佈署以及測試了一個企業級的 **Web service.**

範例 #4. 建置企業層級的網站應用程式

爲了協助 Avitek 能將 Order Entry System 與企業夥伴相連結，在之前的範例中，你已經使用 WebLogic Workshop 的功能來建立強大、企業層級的網路服務。現在，將你的注意力放在網站應用程式上，以瞄準無數潛在的電子商務客戶。很幸運的，由於 BEA WebLogic Workshop 的統合開發架構，經過之前的範例，對於建立客製化的控制元件和網路服務應用程式已經相當熟練，所以開發一個網路應用程式同樣利用的這樣簡化的程式模型。在開始這個單元範例之前:

- 如果之前的範例還在執行的話，請關掉 Workshop Test Browser，並返回 Workshop 的開發環境，儲存並關掉開啓的檔案包括 *orderEntryService.jws* ，以清除工作區。
- 在 Appication 視窗裡，找出 Controller.jpf 檔案並點擊兩下來開啓它。

Java Page Flow 技術

了解 WebLogic Workshop 8.1 創新的 Java 程式開發技術 — Java Page Flow，是使用 BEA WebLogic Workshop 建置伺服端應用程式，並配合動態的 JSP/HTML 使用者介面的必要條件。一個 Java Page Flow 簡潔地表達一些互有關聯的 JSP 網頁和 Java 控制項的集合。透過瀏覽器，一個 JSP 網頁提供一個便利的方法，使你能夠動態地呈現資料。JSP 之所以可展現動態內容是因爲它們並非純粹由 HTML 所組成；相對地，它們包含了可呈現動態內容的 Java 程式碼。而且，Java 的控制類別(由檔案字尾爲.JPF 表示)提供了導覽上和行爲上，橫跨相關網頁的控制。這些部分會一起運作以協助建置企業層級、標準的網站應用程式。

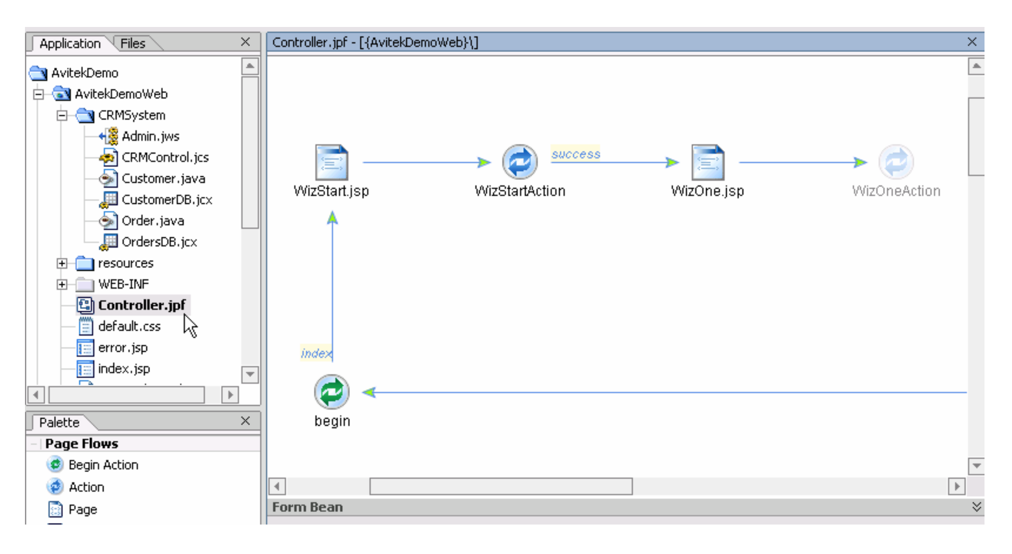


圖 43. Flow view 提供了網站應用程式視覺化的描述

透過預設 Flow View 的呈現，你應該可以看到在你面前的 *Controller.jp* 檔案。Controller.jpf 檔案本身是一個類別，它包含你必須完成網站應用程式的商業邏輯。同時它也包含了必要的方法和程式註解以定義這個網站應用程式所需要的行爲，像是在網頁與網頁間的導覽流程，被網頁執行的 actions(ie 商業邏輯)，以及在此應用程式中共享的網頁資料。Page Flows 可以協助控制元件存取現存的商業邏輯和後端資源。重要的是，Java Page Flow 提供了一個方式，可以細膩地從導覽層的控制邏輯，分離出呈現層的程式碼。呈現層的程式碼可以放在它本來的位置 – 也就是 JSP 檔案；而導覽層的控制邏輯可以建置在每一個 Java Page Flow 裡的組態檔，也就是 JPF 檔案。這樣

的功能性分割給予了開發團隊成員一個好處，因爲他們可以在單一的 JPF 檔修正商業邏輯，而不用找尋每一個 JSP 檔去做多次的更新。

由於 Avitek 的需求，你的目標是建置一個 Web 介面的 Order Entry System 以讓客戶能夠線上訂購產品。當然，你也需要蒐集必要的客戶資訊，掌握客戶端的使用者是新的或是現存的客戶。圖 44 描繪出客戶使用 Avitek Order Entry 網站應用程式時，會經過的流程：

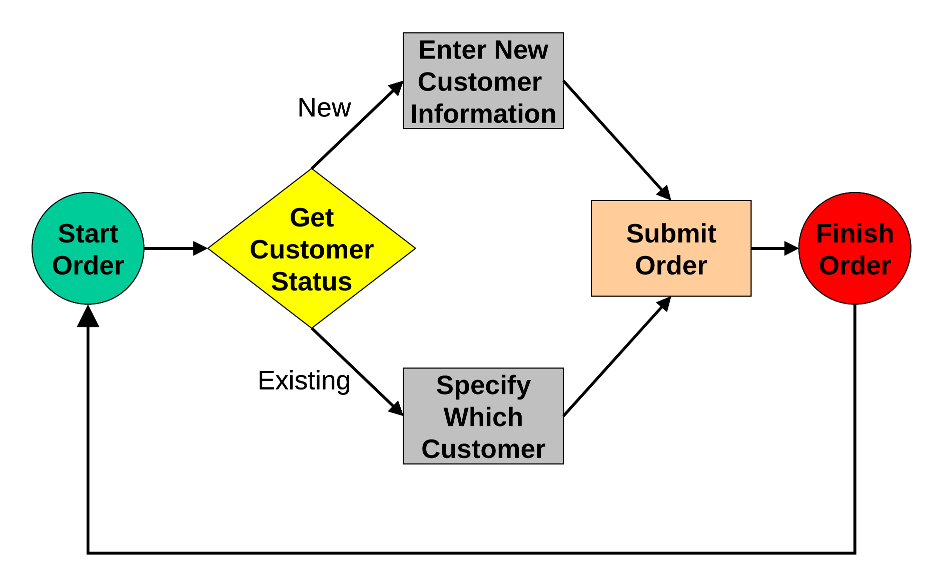


圖 44. Avitek 網站應用程式的處理流程

爲了開發此範例的網站應用程式，你會利用在 Exercise #2 所建的 CRMControl 來掌控 Avitek 的後端資料庫的介面。另外，此網站程式的其他部分已經事先定義好了。事實上，Controller.jpf 的 Flow View 提供了先前定義的網頁與 Action 一個視覺化的描述，組合的結果就如同**圖 44**。

爲了完成 Order Entr 的網站應用程式，你將需要添加遺漏的 Action 及提供必要的商業邏輯。特別的是，你必須要留意在 Flow View 之中，WizOne.jsp 圖示和 WizTwoNew.jsp/WizTwoExisting.jsp 圖示之間遺漏的連結。

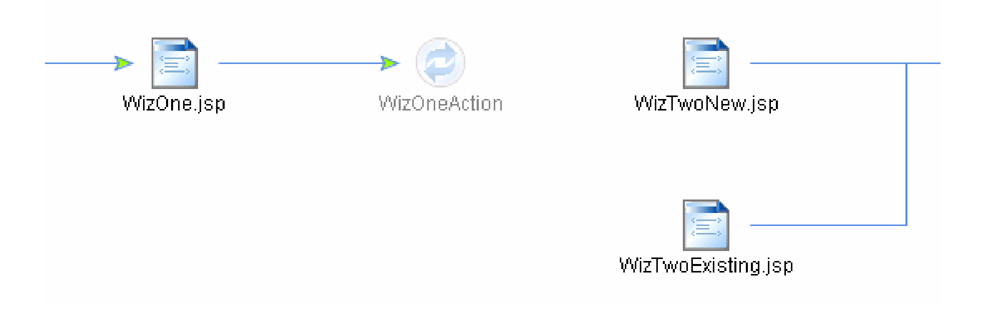


圖 45. 注意在 Java Page Flow 中遺漏的部分

這些遺漏的連結代表著此網站應用程式中，你必須注意的地方。就定義而言，WizOne.jsp 能讓使用者指明自己是 Avitek 的新客戶或舊客戶。

- 在 Flow View 裡，找到 WizOne.jsp 的圖示，並點擊它，你可以看看這個網頁是如何定義的。當你想要結束時，關掉網頁右上方的 X 按鈕。



圖 46. WizOne.jsp 讓使用者說明自己的身分

藉由使用者在 radio button 的選擇，網站應用程式需要處理一些商業邏輯，並且決定使用者應該被導向的路徑。這功能會實作在 WizOneAction，當使用者在 WizOne.jsp 傳送一個請求時，WizOneAction 這個方法會在 Controller.jpf 裡被自動呼叫。

- 在 Flow View 裡找出 WizOneAction 的圖示，它位於 *WizOne.jsp* 圖示的旁邊，而且比其他圖示更呈模糊狀態。爲了開始著手進行，首先在 *WizOneAction* 圖示上按右鍵，選擇 **Create**。

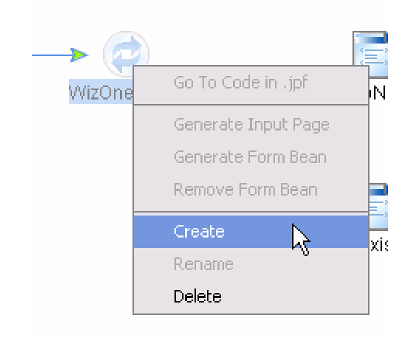


圖 47. 利用 pop-up 選單命令啓動 WizOneAction

下一個步驟會定義與 WizOneAction 相關聯的 Form Bean。一個 Form Bean 是在網頁(即 WizOne.jsp)和它相關的 Action (即 WizOneAction)之間溝通資料的一種機制。因爲 WizOne.jsp 允許使用者去確認自己是新客戶或舊客戶，所以你需要這個與 WizOneAction 相關聯的 Form Bean 來掌握這個資訊。

- 對 WizOneAction 圖示按右鍵，選擇 **Generate Form Bean**，將自動跳出 Form Bean Editor 視窗
- 在 *Form Bean Editor* 裡 ，按一下 **New Field** 圖示，或是在視窗下方的表格中點擊第一列，來產生一個新的欄位。接下來，命名這個新的欄位爲 customerType，並且讓此欄位的資料型態爲預設的 String。完成以後，點擊 **OK**。

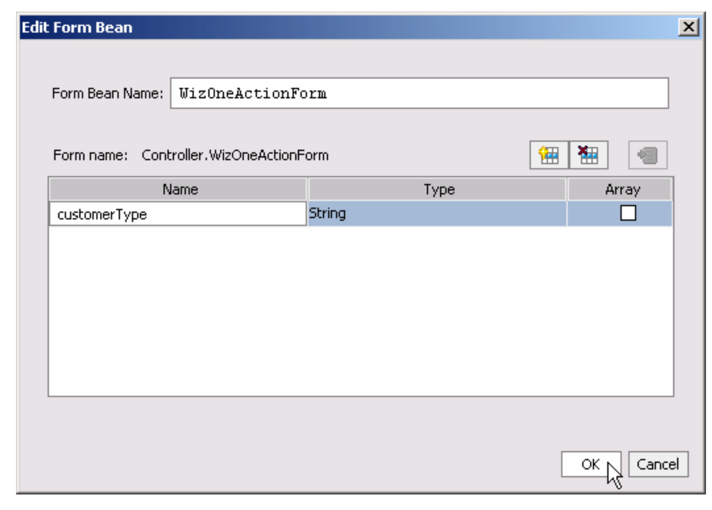


圖 48. 新增一個 String 欄位到 Form Bean 以取得使用者的狀態

此時，這個 Form Bean 已經完成，接下來它將會傳送 customerType 的值到 WizOneAction。藉由這個值(所指的是新客戶或舊客戶)，WizOneAction 將會決定使用者該導向至哪一個網頁去。根據這個流程，你需要透過視覺化的操作，在此網站應用程式的流程裡，將新建的 WizOneAction 與接下來的網頁相互連結。

- 在 *WizOneAction* 與 *WizTwoNew.jsp* 間描繪出一個連結，你只要在 *WizOneAction* 圖示的右邊按一下(此時滑鼠的指標應該變成一個底下有箭頭的連結，如同**圖 49)**並且拖曳的出一個連結(持續按住滑鼠)到 *WizTwoNew.jsp*，接下來再將新產生出來的連結，重新命名為 **new**(在黃色小方塊裡)。



圖 49. 視覺化的拖曳來連結 Flow View 裡的網頁

- 同樣的操作方式，再拖曳出一個 *WizOneAction* 到 *WizTwoExisting.jsp*,之間的連結，並且重新命名為 existing。

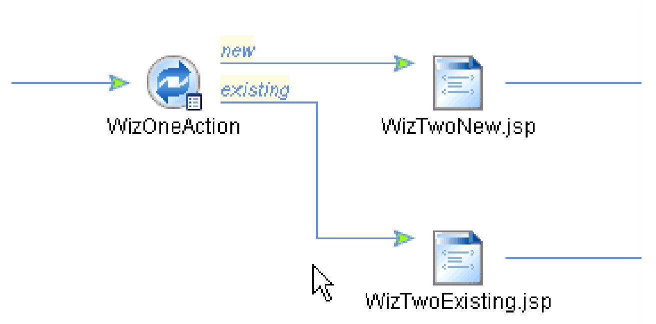


圖 50. Flow View 呈現特定的導覽邏輯

就如同你看到的，Flow View 提供了一個簡單又直覺的介面，來說明和檢視整個 Order Entry 網站應用程式的導覽流程。

➢ 在 WebLogic Workshop 裡的網站應用程式，你可以點擊主畫面下方的 Action View 的標籤，進入 Action View。

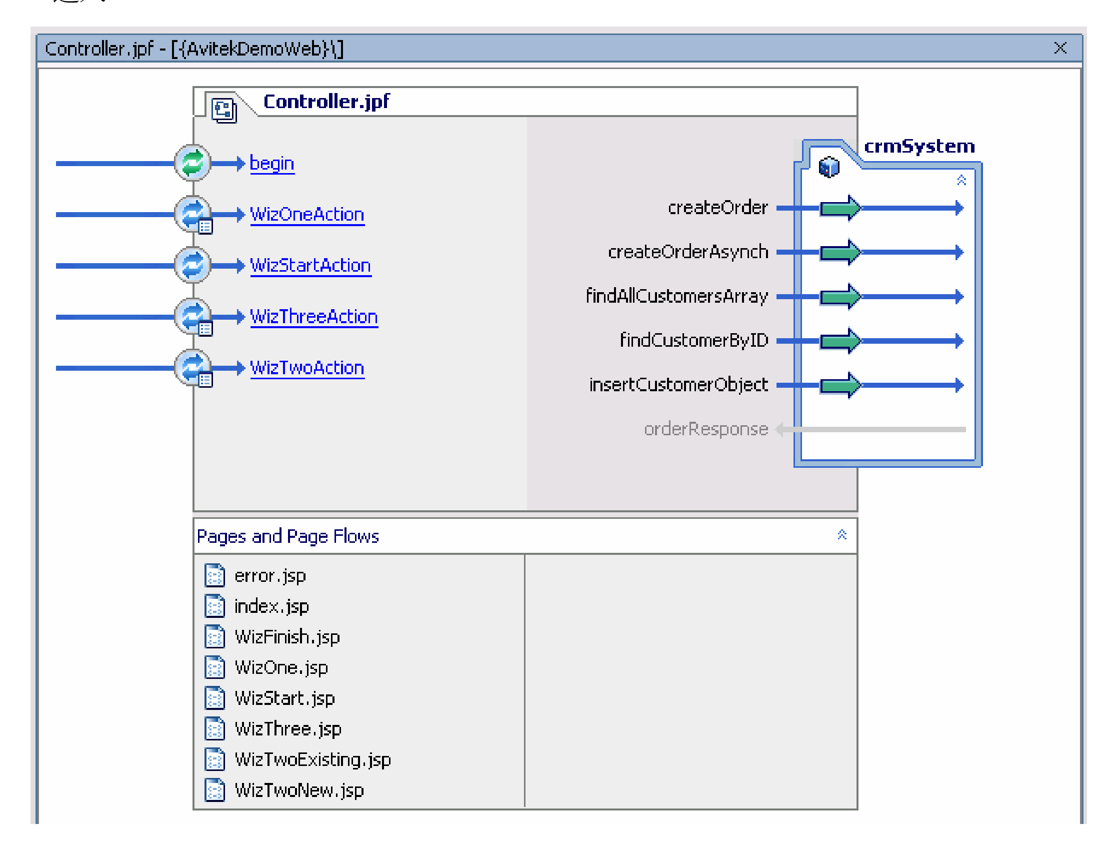


圖 51. Action View 呈現 Controller.jpf 裡相關的方法和資源

Action View 看起來會與你在開發 custom control 與 Web service 時的 Design View 相似：在右邊包含方法，而在左邊有相關資源。以網站應用程式而言，這些方法代表著控制應用程式行為的 Action。現在，點擊任一個 Action 的名字，你將可以在 Source View 看到對應的程式碼。

➢ 在 Action View 中，點擊 *WizOneAction* 的超連結以將畫面轉換到 Source View，並且檢視此 Action 的 Java 程式碼。

```
/**
 * @jpf:action
 * @jpf:forward name="new" path="WizTwoNew.jsp"
 * @jpf:forward name="existing" path="WizTwoExisting.jsp"
 */
protected Forward WizOneAction(WizOneActionForm form)
{
    return new Forward("success");
}
```

圖 52. 程式註解以顏色標明，增強可見度

注意此程式碼的注解，在 WizOneAction 的方法定義上面，有一個符號@jpf，如**圖 52**。這些符號都是當你在 Flow View 中描繪出視覺化的連結時，自動地被開發環境嵌入進去的

- 在 *WizOneAction* 方法裡，你需要用下列的程式來取代預設的單一程式行，以決定使用者的型態，然後執行適當的 if/else 邏輯。當你完成以後，這個方法應該如下圖所示：

```
/**
* @jpf:action
* @jpf:forward name="existing" path="WizTwoExisting.jsp"
* @jpf:forward name="new" path="WizTwoNew.jsp"
*/

public Forward WizOneAction(Controller.WizOneActionForm
   form)
{
      if (form.getCustomerType().equals("existing"))
            {
            custInfo = crmSystem.findAllCustomersArray();
            return new Forward("existing");
            }
      else
      return new Forward( "new" );
}
```

自動化的資料繫結

現在，WizOneAction 已經完成了，你幾乎可以去測試這個網站應用程式，然而現在還有一個更重要的工作：

- 回到 Controller.jpf 的 Flow View，找出 WizTwoNew.jsp 檔案，點擊兩下開啓它。

現在你將會看到 WizTwoNew.jsp 會載入到 WebLogic Workshop's JSP 編輯器。這個編輯器使開發人員設計出複雜、使用標籤和屬性的 JSP 表單，而且支援提供拖曳與所見即所得(WYSIWYG)功能的 Design View。因此在這種情況下，對你而言，WizTwoNew 網頁幾乎是空的，所以你只需要從事必要的附加步驟，以讓網頁能夠從需要存到 Avitek's 資料庫的新客戶中，蒐集到相關資訊。很幸運的，Workshop 簡化了這個流程，可藉由拖曳、資料繫結的表單，自動地將使用者介面與後端資源作連結：

- 在 Data Palette 中找出 *WizTwoAction* 的圖示，並且將它拖曳到 Design View 的 *WizTwoNew* 網頁。

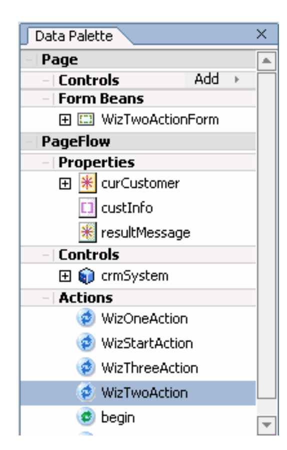


圖 53. Data Palette 聚集 Java Page Flow 的相關元素

- 這樣的動作會帶出 Form Wizard.視窗。
  - 讓 Action 為預設的 WizTwoAction，選擇 Next。



  - 你會看見 *WizTwoAction 的* Form Bean 中預先定義的資料欄位。將 *custID 欄位取消勾選*，代表著新客戶不應該輸入他們自己的客戶代號，因為此欄位值是系統自動產生的。接下來按 Next 按鈕。

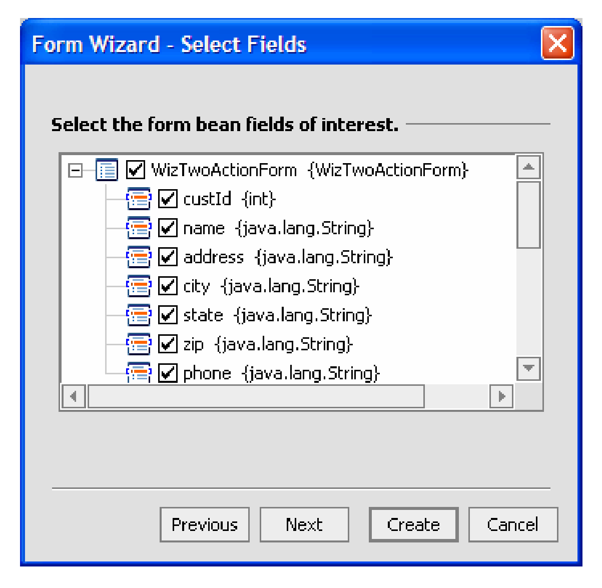


圖 54. 使用 Form Wizard 選擇適當的欄位呈現在表單上

- 接下來，From Wizard 跳出 *Adjust Field* 對話框，在這你可以調整欄位出現的位置，還有用來蒐集資料的輸入介面型態(textbox, checkbox 等)。完成之後，按下 **Create**。

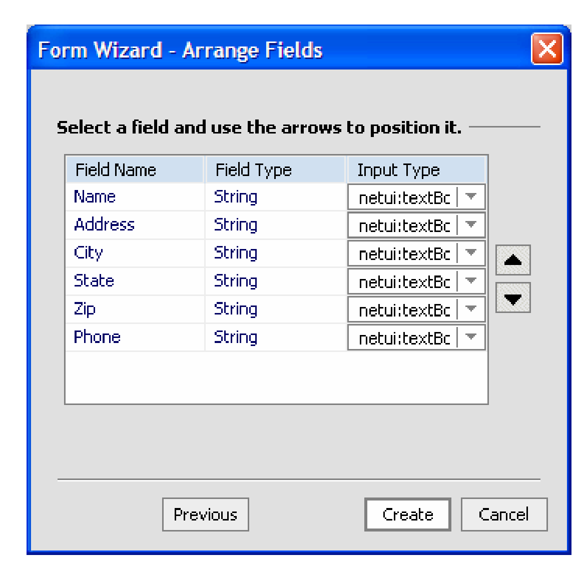


圖 55. 為表單調整合適的欄位位置

JSP 編輯器會更新 WizTwoNew.jsp 網頁，呈現出你剛剛附加變更的結果，就如**圖 56**。

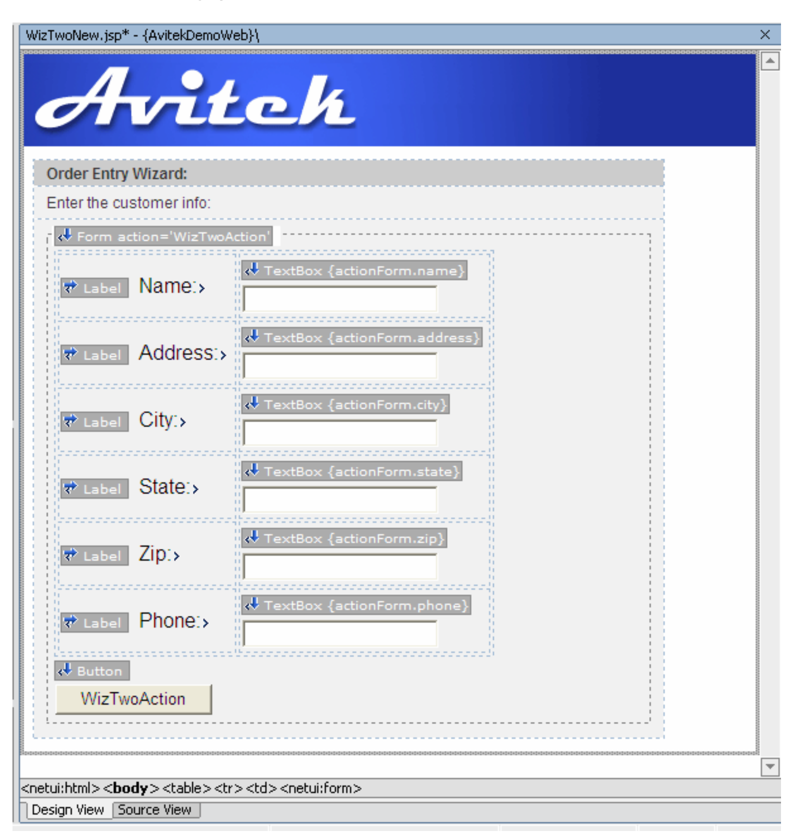


圖 56. WebLogic Workshop 8.1 提供 WYSIWYG 功能的 JSP 編輯器

WebLogic Workshop 8.1 自動地產生你剛剛在 Form Wizard 設定的資料繫結表單。由此可見，運用 WebLogic Workshop 建置一個包含資料庫存取、網路服務或客製化商業邏輯的網站應用程式，竟是如此的容易。資料繫結的功能，可藉著外部的標籤庫運作，並且支援更為複雜的資料像是

grids、lists、repeating elements 等。除了資料管理，WebLogic Workshop 8.1 會自動掌控 session、狀態管理及其他複雜的細節，所以開發人員可以把焦點放在商業問題上。

自動化的部署與測試

這個部分將完成 Avitek Order Entry System 網站應用程式所需的步驟，所以現在你可以測試你所開發的應用程式。再一次地，你可以利用 WebLogic Workshop 的自動化部署級整合性的測試，使得循環開發更為輕鬆容易。

- 儲存並關閉 *WizTwoNew.jsp* 檔，回到 *Controller.jpf*。按下 Start 按鈕(或是 **Debug → Start without Debugging)**，將會自動地部署 Order Entry 網站程式，並且跳出 Workshop Test Browser。再這裡必須注意的一點，對於任何的網站應用程式，第一次載入 JavaServer Page (JSP)會比以後的測試花更多的時間，因為呈現層的程式碼會在第一次載入時被編譯。
- 從 Workshop Test Browser 一開始的頁面，按下 **Start Entering An Order!** 啟動 Avitek Order Entry System。

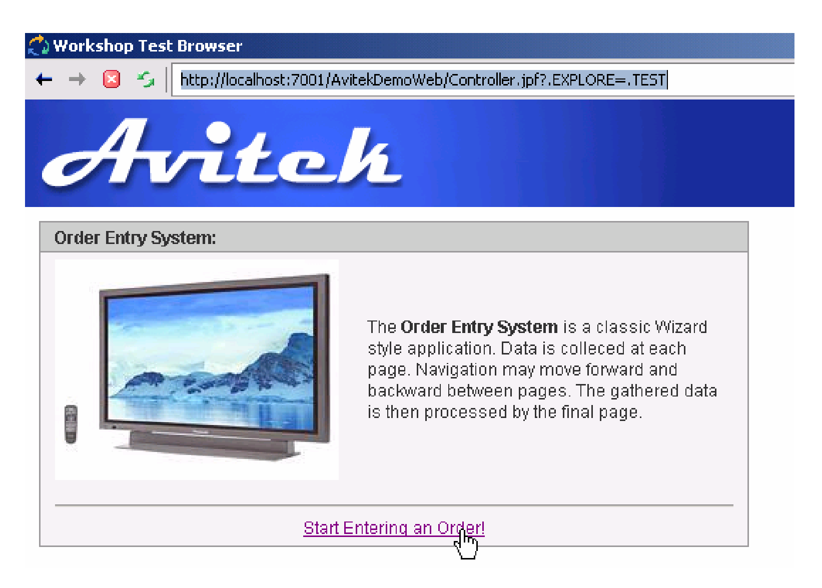


圖 57. Avitek Order Entry System 的登入頁面

- 由於你是第一次瀏覽，用 radio buttons 確認自己為新客戶，完成之後按下 Next。

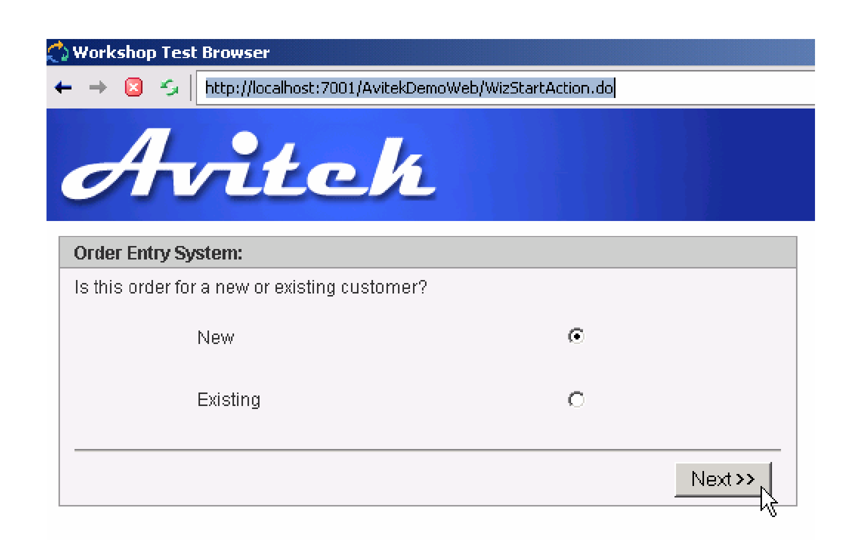


圖 58. 此應用程式利用簡單的使用者介面元件，來蒐集顧客的資訊

- 由於你之前在 *WizOneActionz* 方法鍵入的商業邏輯，你會被正確地導覽至“new customer”的路徑，並且可以輸入你的相關資訊。在這裡可以看到你在 *WizTwoNew* 建置出來的資料繫結表單。填入如**圖 59** 的欄位資訊，然後再按下 **WizTwoAction** 按鈕。



圖 59. 在此範例中的新客戶，此應用程式蒐集相關的資訊

- 接下來你可以指定訂購產品的數量，完成之後按下 Next，將會看到確認資訊的網頁。

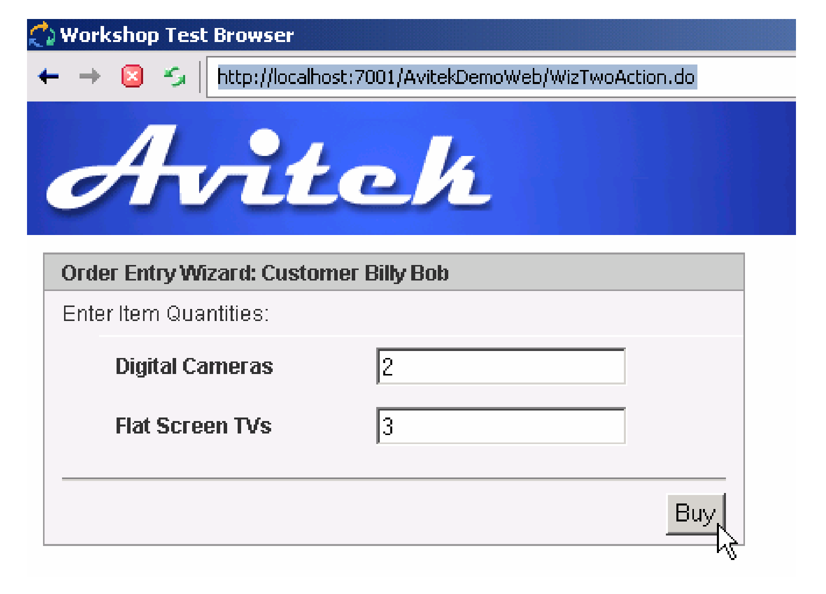


圖 60. 此應用程式現在蒐集訂單資訊

對於所有的新客戶，來到 Avitek's 網站訂購電視和數位相機，是如此的容易!確定客戶的資訊是正確掌握的，接下來，讓我們再一次造訪 Order Entry System：

- 從確認網頁按下 **Start Another Order**

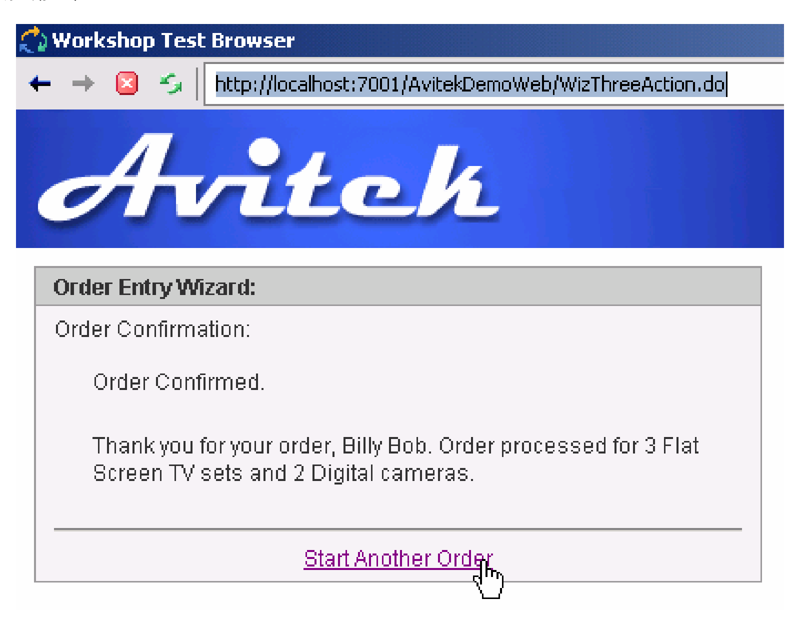


圖 61. 此應用程式確認最初的訂單，並允許使用者開始另一筆訂單

- 這步驟將會回到一開始的首頁，跟之前一樣，**Start Entering An Order!**

- 在這一次，確認自己的身分爲 **Existing** 客戶，然後按下 Next。

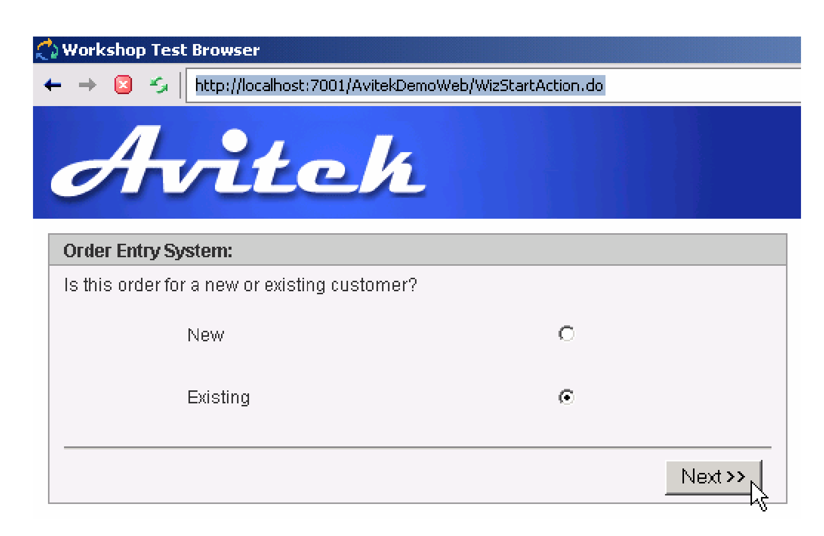


圖 **62.** 在這一次的操作中，指名自己為 Existing 的客戶類型

- 現在，你可以檢視 Avitek 客戶資料庫裡所有的客戶列表。你應該可以看到一筆之前加入的資料-- Billy Bob。此外，你也可以看到 Bob Smith 和 John Doe 這兩筆記錄，它們是前後分別由 *JavaOrder.xml* 與 *PurchaseOrderMessage.xml* 在 Exercise #3 傳送的。

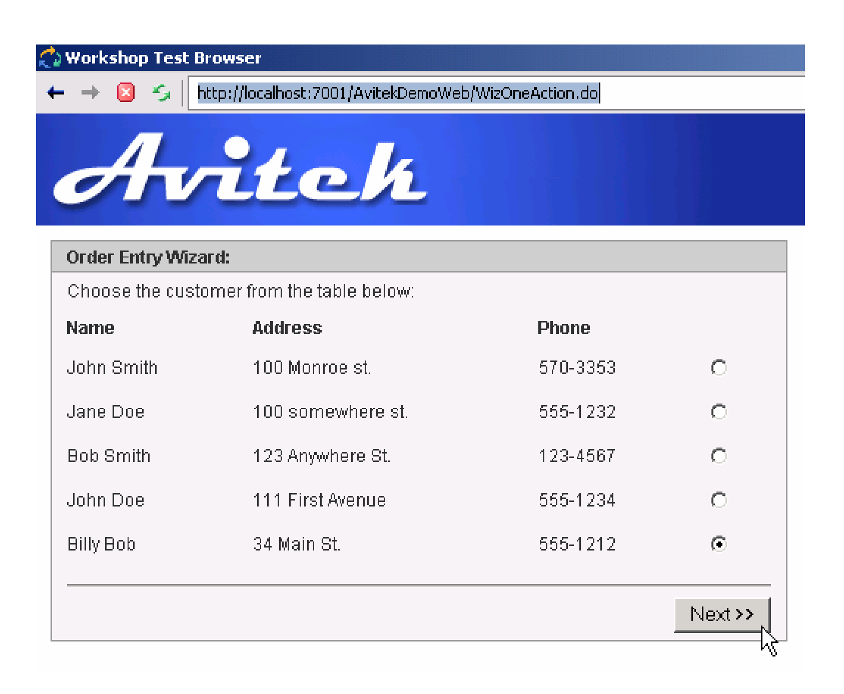


圖 63. 此應用程式呈現 Customer Database 裡所有的紀錄

恭喜你，Order Entry System 網站應用程式已經完成了!BEA WebLogic Workshop 8.1 讓使用者介面的建置容易上手，因為你只要專注在邏輯、流程導覽與資料上。WebLogic Workshop 8.1 的執行期框架幫你掌控低階的細節，並幫助你提高生產力。

在之前的範例，你已經成功建置了標準規格、企業層級的應用程式，使 Avitek 更敏捷、更有競爭力。重要的是，你不需要任何 J2EE，或是物件導向的觀念，就可以達成這樣的目標；此外，你將比沒有 BEA WebLogic Workshop 8.1 協助的開發者，花更短的時間完成這項工作

範例#5：用 JAVA 控制項方式建立網頁應用程式

之前的範例中，你已經透過企業邏輯、流程控制及資料繫結的方式來完成”訂單登錄系統”的網頁應用程式。但是你若要快速且具體開發網頁應用程式的話，例如：Avitek 網站提供資料庫管理者存取”Orders”資料庫以及執行簡易的資料庫管理工作(紀錄的新增、查詢、修改及刪除)。那麼，Workshop 就是能夠從 Java 控制項或資料庫的表格中產生相對於”CRUD”功能的網頁應用程式，輕易完成上述複雜的功能需求。

- 開始之前，你需要建立新的網頁應用程式。建立之後，儲存並關閉所有的檔案。在”*AvitekDemoWeb*”檔案夾上點擊滑鼠右鍵，並選擇 **New->Page Flow**。
- 出現 Page Flow Wizard 對話盒之後，在 Page Flow Name 欄位填入 *OrderAdmin*，此時，Controler File 欄位也會自動修改。完成後，再點擊 **Next** 按鈕。

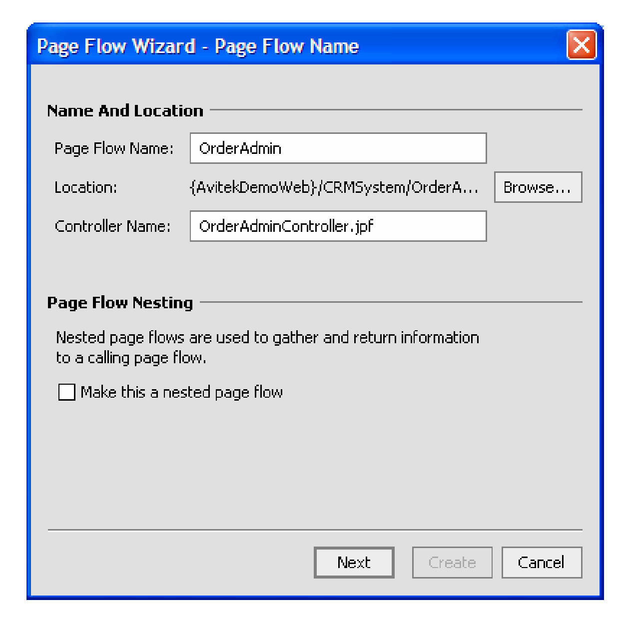
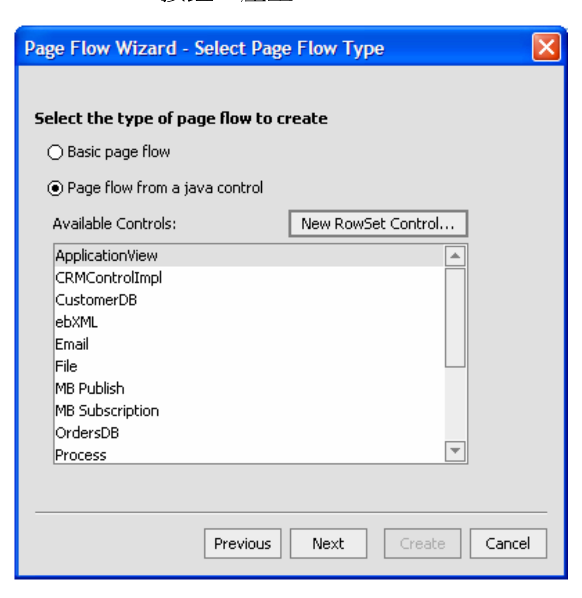


圖 64. 建立名為”OrderAdmin”的 Page Flow

- 接下來的步驟將帶領你快速建立新的 Page Flow 或根據既有的 Java 控制項建立 Page Flow。使用 Database 控制項(須事先定義)，選擇 **Page Flow from a Java Control** 欄位並點擊 **New Rowset Control** 按鈕，產生 Database Control Wizard 對話盒。



- 在 Control Name 欄位填入 OrderRS 及保留 Data Source 欄位的預設值 cgSampleDataSource，此步驟將 WebLogic Platform 中嵌入一個新的資料庫表格。完成後，點擊 **Next** 按鈕。

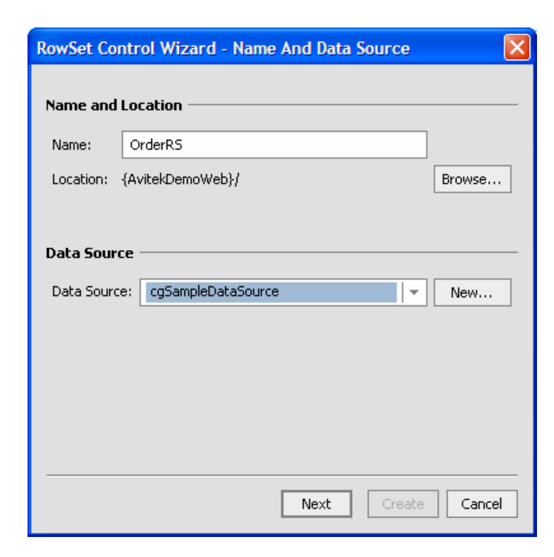


圖 65. 使用 Use the Database Control Wizard 建立及設定新 Control

- 允許查詢及修改資料庫表格，所以保留單選方塊的預設值 Query and update a data table。並在下方 Schema 欄位中選取 **WEBLOGIC** 以及 Table 欄位中選取 **ORDERS**。.

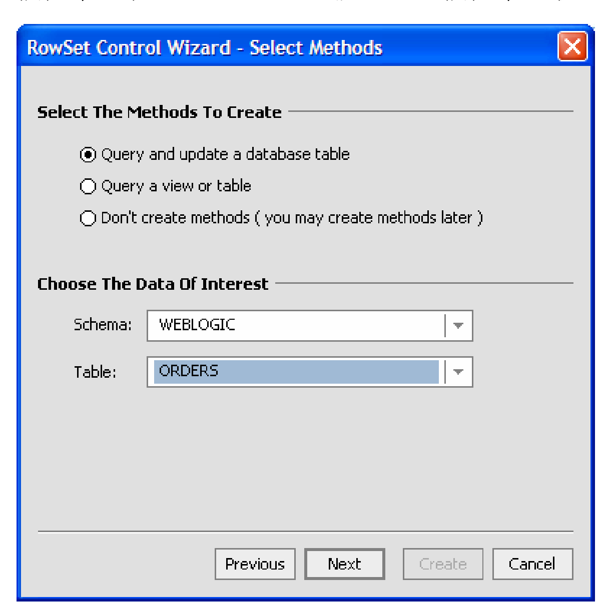


圖 66. 選擇 WEBLOGIC schema 與 ORDERS table

- Database Control Wizard 的下一步驟，你可以看到 Orders 表格所有的欄位，此時，保留所有預設值，直接點擊 **Next** 按鈕到下一步驟。
- 此個步驟中，你可以直接使用資料庫自動產生的主鍵(預設選取的)，點擊 **Create** 按鈕完成 Database Control Wizard 的設定。

- 回到先前建立 Page Flow 的 Control 對話盒，選取名為 ***OrderRS* 的 *Control*，並點擊 *Next* 按鈕**。

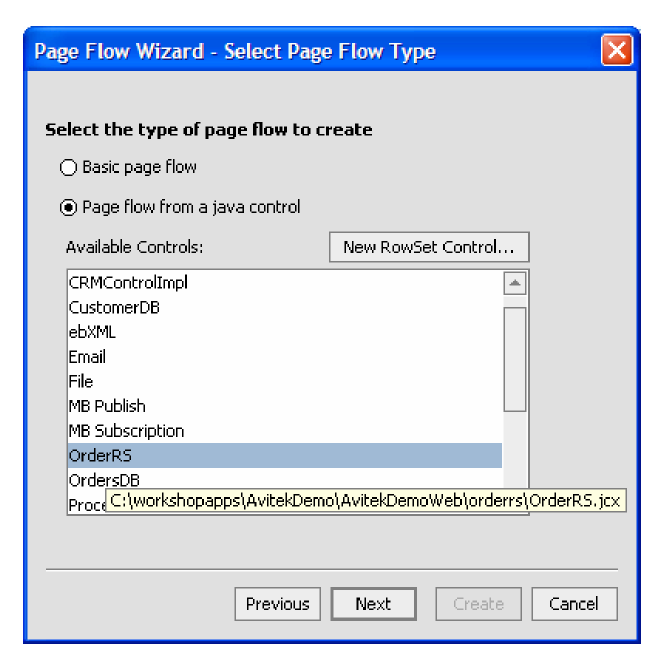


圖 67. 新增一個 Java Page Flow, 取名為 orderAdmin

- 此個對話盒中，你可以選擇 OrdersAdmin 應用程式所允許的動作。但是，為了配合實際狀況，所有訂單是由訂單登錄系統所產生，資料庫管理員不該有新增訂單的權限，所以取消 Insert New Rows 勾選方塊，如**圖 68** 所示。完成後，點擊 **Create** 按鈕。

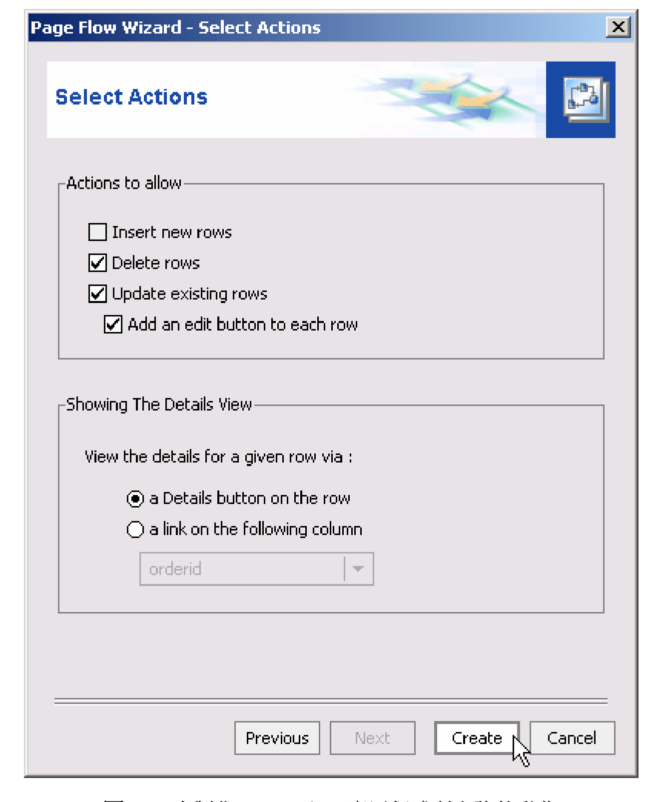


圖 68. 客製化 Page Flow 應用程式所允許的動作

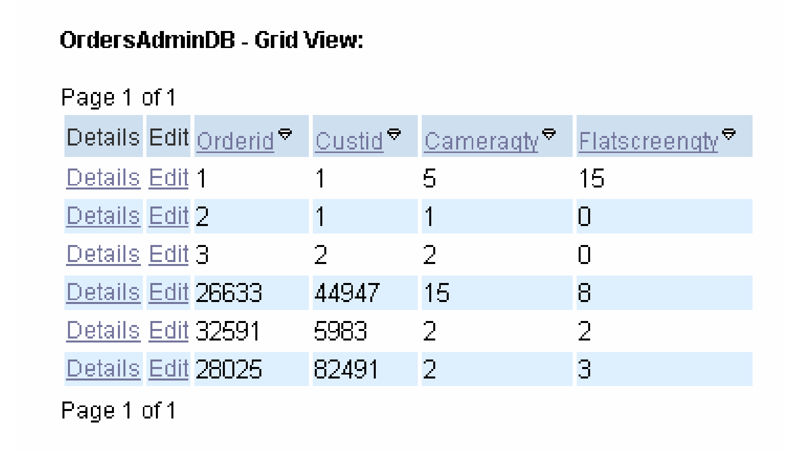
你將會在 Flow View 中看到剛剛建立的 OrderAdminController Page Flow：

圖 69. WebLogic Workshop 8.1 會自動產生相對應的網頁、流程及動作

如同你看到的，BEA WebLogic Workshop 自動產生所需要的 JSP 網頁、導覽指示以及相關動作，符合你所要管理的 Orders 資料庫表格。當然，程式開發者可以輕易地擴充此應用程式，但是由於剛剛建立的 Page Flow Wizard 是一個完全功能開放的網頁應用程式，所以任何經過核可通過的 Avitek 使用者可以針對 Orders 資料庫表格進行資料庫管理動作。

不只建立資料庫管理而已，你也已經建立合乎企業級類別的應用程式，即符合 MVC(模式-景觀-控制者)架構及達到 Struts 的標準。Workshop 使程式開發者很容易完成原本複雜工作，因此，即使你不是這些技術的專家，也可以開發出符合 J2EE 架構的應用程式。

➢ 檢視剛才建立好的網頁應用程式，可以點擊 **Start** 圖示來部署應用程式及啟動整合的測試畫面。目前 Orders 資料庫表格所有的紀錄將會出現在第一個畫面中，如**圖 70** 所示。



OrdersAdminDB - Grid View:

Page 1 of 1

Details	Edit	Orderid	Custid	Cameraqty	Flatscreenqty
Details	Edit	1	1	5	15
Details	Edit	2	1	1	0
Details	Edit	3	2	2	0
Details	Edit	26633	44947	15	8
Details	Edit	32591	5983	2	2
Details	Edit	28025	82491	2	3

Page 1 of 1

圖 70. Grid View 顯示出 Avitek Orders 資料庫表格的所有紀錄

➢ 如果你點選某一筆紀錄的 **Edit** 連結，將會出現紀錄的 Edit View，允許紀錄的修改和刪除。

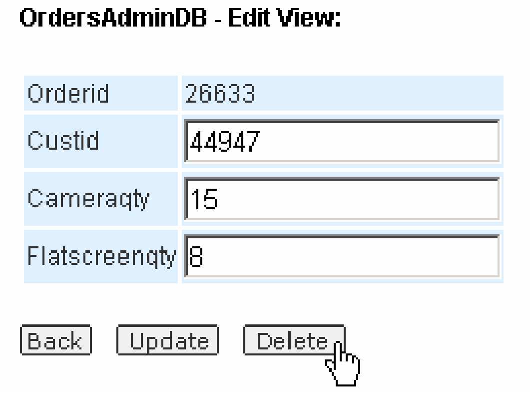


圖 71. 應用程式讓管理者輕易地修改或刪除目前訂單紀錄

恭喜你！在短短的時間之內，你已經完全熟悉標準資料庫的網頁應用程式的核心功能。

範例 #6: 使用 XMLBEANS 來處理 JAVA 中的 XML

當你建置範例 3 的 Order Entry Web service，你使用了 Workshop 的 XQuery 對應工具來處理 XML 文件與那些用來處理資訊的 Java 資料物件之間的轉換。XQuery 對應方法是名符其實的鬆散耦合 –區分了服務的 公開約定(即 schema 定義)和 Java 的內部實作與 BEA 也提供了另外一項鬆散耦合的 XMLBeans 技術與。XQuery 對應互相配合。

XMLBeans 是一項創新的技術，藉由提供方便的、以 Java 物件爲基礎的觀點的 XML 資料(Java object-based view of XML data)，但又不會失去 XML 架構本身的好處，能讓開發者輕鬆地存取操作 Java 中的 XML 資料和文件。使用 Java 類別的觀點可以自動產生適當的 schema 定義，如同那些你已經做過的之前範例一樣。下面介紹如何在 Workshop 中使用 XMLBeans：

- 清除 workspace，然後在 Design View 重新開啓 *orderEntryService.jws* 檔案。
- 由於下面的動作將會取代之前實作的 XQuery mapping，你需要刪除之前對檔案的編輯。最簡單的方法就是利用 *CreateOrderAsynch* 這個方法(method)，然後在 Property Editor 頁籤下 *Parameter-XML* 欄位的 *XQuery* 屬性，來啓動 *XQuery Editor*。

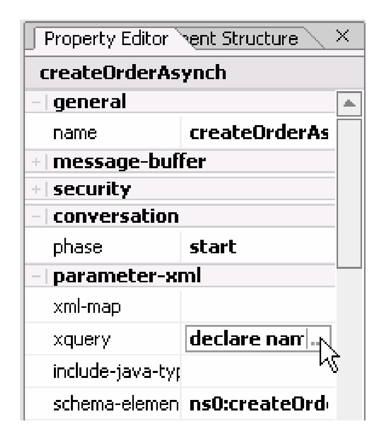


圖 72. 使用 Property Editor 來重新啓動 XQuery Mapping

- 接下來，按下 **Delete** 按鈕來移除你在範例 3 所建立的 transformation。接著按下 OK 來回到 Design View。
- 接下來要在 *createOrderAsynch* method 組合 XMLBeans，只需要按下他的超連結名稱來轉換到 Source View。

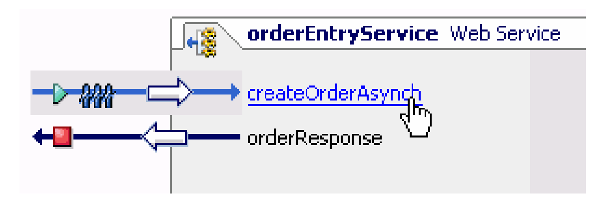


圖 73. 按下該方法的超連結來轉換到 Source View

- 修改 *createOrderAsynch* method 的宣告，讓它從原先接受 *Customer* 和 *Order* 這兩個參數，變成爲接受 *PurchaseOrderDocument* 這一個參數。而程式碼底下的提醒修改線將會如下圖一樣出現：

```
public void createOrderAsynch(PurchaseOrderDocument podoc)
```

注意當你輸入這個新的變數型態，你會看到一個跳出視窗提醒你這個變數需要 import 新的類別。

```
/**
 * @common:operation
 * @common:message-buffer enable="true"
 * @jws:conversation phase="start"
 */
public void createOrderAsynch(PurchaseOrderDocument podoc)
{
    cRMControl.createOrderAsynch(c,o);
}
```

org.openuri.easypo.PurchaseOrderDocument? Alt+Enter

圖 74. WebLogic Workshop 8.1 在 Source View 中包含了方便的 Auto-Import 功能

這是因爲 Workshop 爲了增進建置模組化應用程式生產力的 auto-import 特性。按下 **Alt+Enter** 來自動 import 正確的 *org.openuri.easypo* 套件(這個名稱被定義在範例 3 匯入的 schema 檔案內)。當內建的 XMLBean 編譯器匯入與同步處理時，Workshop 會自動新建一個 PurchaseOrderDocument 的 Java 類型。這個 Java 類別包含一些變數和方法，同時提供這個以 XML 爲基礎的 purchase order 文件完整的存取。

此時你有兩個選擇：一個方法是建立一個新的控制項並把 PurchaseOrderDocument 當作它的參數；另一個是你僅需要從 PurchaseOrderDocument 建立與植入一個 Order 與一個 Customer 的資料結構，如此一來你能夠重複使用在 CRMControl 內已存在的 createOrder 方法。利用後者可以達到簡化的目的。

➢ 插入下面的程式碼來，以使用 XMLBeans 來從 XML 中得到必要的資訊，以滿足 customer 和 order 資料結構中的重要欄位。完成後，這個 createOrderAsynch 方法將會如下所示：

```
/**
 * @common:operation
 * @common:message-buffer enable="true"
 * @jws:conversation phase="start"
 */
public void createOrderAsynch(PurchaseOrderDocument podoc)
{
    Customer c = new Customer();
    Order o = new Order();
    PurchaseOrder po = podoc.getPurchaseOrder();
    c.name = po.getCustomer().getFirstname() + " " +
      po.getCustomer().getLastname();
    c.custId =
      po.getCustomer().getCustomerNumber().intValue();
    o.custID = c.custId;
    o.orderID =
      po.getOrderData().getOrderNumber().intValue();
    o.cameraQty =
      po.getOrderData().getCameraOrder().getQuantity();
    o.flatScreenQty =
      po.getOrderData().getTvOrder().getQuantity();
    cRMControl.createOrderAsynch(c,o);
}
```

當輸入這些文字時，你有可能會遇到 Auto-Import 的訊息，這時你可能需要設定 Workshop 去 import *CRMSystem.Customer* 和 *CRMSystem.Order*。

- 要測試這個修改過的 Web service，按下 **Start** 圖示。
- 按下在 Workshop Test Browser 的 **Test XML** tab。
- 在 Application Window 你將會看到 *PurchaseOrderXBean.xml* 檔案，複製它到 test browser 的 SOAP body text-box。按下 **createOrderAsynch** 按鈕來傳送訂單訊息。
- 就像前面一樣，你必須按下 **Refresh** 來擷取非同步傳輸的相關訊息。當 callback.orderResponse item 出現在 Message Log 時，可以看到 order confirmation message 在 Client Callback。

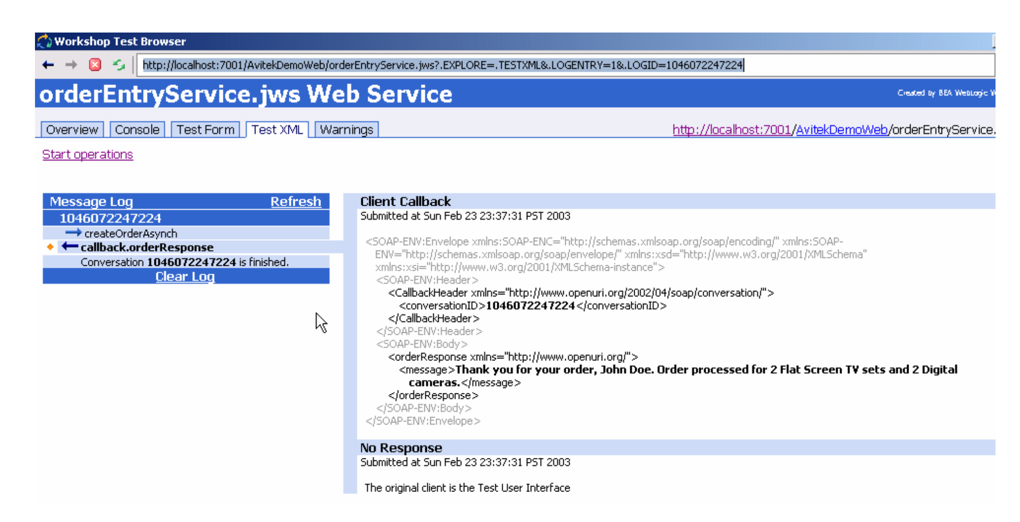


圖 75. test client 追蹤非同步的訊息對話並顯示 order 的確認(acknowledgement)

如同你所見，Workshop 使用 XMLBeans 來讓處理 XML 變的容易。這個情境只描述出解析 XML 資料，你可以相同地使用一組"setter"函式來擴充與修改 XML 訊息。重要的是，Workshop 自動地保護所有 XML document 的變動，而這 XML document 是依照預期的結構而傳送的 schema。當使用 XMLBeans 來新建這些低耦合度應用程式時，這種 XML 保護有著相當大的好處。

要如何決定哪裡適合使用 XQuery 對應或 XMLBeans 在你的應用程式中，一個重要的考量是處理的種類：是 end-point 或是 way-point。若是 end-point，程式是一個 XML 的 end point，它的責任是取得 XML 內它所需要的的資訊，或是新建一個 XML 文件。這個應用程式可以使用文件來執行複雜、程序性的事情，並且可以摒棄一些不需要的 XML 文件。另一方面，在 way-point 的處理情況下，程式把一個 XML 文件對應到它的一個程式流程。在這種情況下，程式並不會是"lossy"，因爲它在商業流程上不知道那個函式優先或跟隨在之後，或者預期接下來是那個註解。在這種情況下，整份 XML 結構的保護是關鍵的。如同你所想的，這些 way-point 應用程式特別適合 XMLBeans，反之，end-point 應用程式可能只需要使用 XQuery 對應。通常兩者都可以使用，而取決於開發者的決定。

恭喜你！你已經完成了所有的在快速上手指南中的範例。

BEA WEBLOGIC WORKSHOP 8.1™背後的奧妙

可以利用 BEA WebLogic Workshop 8.1™簡單輕鬆地來建置企業級的應用程式只能算是個開端。即使您沒有寫過任何一行 J2EE 的程式碼也沒有關係，您所實作出來的應用程式經由目前執業界牛耳的 BEA WebLogic Server 來執行，依然可以享受到 J2EE 龐大且豐富的資源，例如 entity beans, stateless session beans, JDBC 連結, and JMS 佇列。

由目前的使用經驗中一路走來，相信您已經多少可以感受到利用 BEA WebLogic Workshop 8.1 實作出可執行、結構化的應用程式，例如客制化的 Java 控制項、Web services 及 Web application 實在不是一件困難的事情。但這裡仍然有一些您還沒有看到的強大功能－所有的相關機構發表的重要功能及 J2EE 平臺上具有的優勢都被納入 WebLogic Workshop 8.1 的執行架構中。WebLogic Workshop 8.1 所實作出來的應用程式因此受益於 EJBs、JMS 佇列及其他先進的功能，而開發人員不一定需要全部熟稔這些複雜的技術。如果您希望了解更多 BEA WebLogic Workshop 8.1 內部的相關訊息，您可以透過 WebLogic Server 管理主控台來查看。

在實作過這六個範例之後，您應該不難發現使用 BEA WebLogic Workshop 8.1 能讓每一位開發人員都可以輕易建置功能強大、標準化的應用程式，同時可利用這款業界翹楚應用伺服器上多項強大的功能，例如叢集、快取、pooling 以及訊息佇列等等在其他地方是只有少數訓練有素的 J2EE 開發者才有辦法使用的技術。有了 BEA WebLogic Workshop 8.1，讓您建置應用程式時，每件東西就變得像一個單純的 Java class 同樣地簡單。

若您沒有 BEA WebLogic Workshop 應用程式開發環境的奧援，即使是想要完成像您在範例中所實作的應用程式，也需要長期累積對 APIs 的專業知識，例如 JNDI、JMS、 及 JDBC 才能辦到。同時也需要通曉 XML 的標準，例如 SOAP 以及 WSDL，其他還有 Struts 架構、MVC 模型、狀態管理(state management)、訊息關聯(Message Correlation)、資料永續儲存及 XML-to-Java 對應等等多樣的技術也必須了解。所幸 BEA WebLogic Workshop 將這些架構中複雜的部分給抽象化，讓開發人員可以不用熟悉所有專業知識就能解決商業上棘手的問題，締造出更迅速的價值時程。

結語

BEA WebLogic Workshop 8.1™在 J2EE 開發領域是一個令人興奮的新發展。透過利用簡易的程式模型來建立強大、標準的應用程式，WebLogic Workshop 8.1 提供 IT 開發組織前所未有的生產力，並且增加了他們的 time-to-value。

透過瀏覽快速上手指南所提供產品維度的概述，你將會體驗一些 BEA WebLogic Workshop 的重要優勢：

讓 J2EE 容易上手和更有生產力：WebLogic Workshop 視覺化開發環境 和(執行期框架)解決了 J2EE 本身的複雜度。WebLogic Workshop 提供簡化的概念，如：控制項、屬性、和事件，讓開發者專注於商業邏輯，而不是底層的細節上。代替了低階 J2EE API 這種需要長時間經驗的開發方式。這意味著已經在企業大量應用的 J2EE，主要將是有視覺化或程序性技能的開發者，而他們的生產力是直接使用低階 J2EE API 的十倍。

降低資訊技術的複雜度：BEA WebLogic Workshop 8.1 提供開發者單一的工具和簡化的程式模型(programming model)來建置與整合在 BEA WebLogic Enterprise Platform™ 8.1 平台上任何型態的應用程式，包含 Web applications, Web services, Portals, 和 Integration applications。這種方法簡化了開發的組織，並且讓開發者能使用他們已知的技術在任何平台專案上工作。WebLogic Workshop 也提供了可延伸的 Java 控制項方法論，讓輕鬆整合已存在的 IT 基礎架構如資料庫、既有的應用程式、以及其他後端資源，讓開發者花較少的時間完成。

增加企業級應用程式專案的成功：BEA WebLogic Workshop 8.1 透過實作可靠、可信賴、以及安全的企業級架構，來減少開發的風險。Workshop 8.1 執行期框架能自動建立符合標準 J2EE 元件的應用程式，來保護你投入的技術與保持最大的彈性。

BEA WebLogic Workshop 8.1 是一個統一、簡化、和可延伸的開發環境，能讓不論是否專精 J2EE 的開發者，在 BEA WebLogic Enterprise Platform 平台上去建置、測試、和部署企業級應用程式。

附錄：BEA WEBLOGIC WORKSHOP 8.1™需求規格

支援平台：

Microsoft Windows XP

Microsoft Windows 2000

Linux – Red Hat Advanced Server 2.1

Sun Microsystems 8 and 9 (只有執行時期)

HP-UX 11.0 and 11i (只有執行時期)

應用程式伺服器：

BEA WebLogic Workshop 8.1 Application Developer Edition 需要使用到 BEA WebLogic Server 8.1。

BEA WebLogic Workshop 8.1 Platform Edition 同樣也需要使用到 BEA WebLogic Portal 8.1 以及 BEA WebLogic Integration 8.1。

進階：產品資訊和開發資源

請參觀BEA dev2dev開發人員的入口網站http://dev2dev.bea.com，該網站擁有許多資源，包含產品訂購、最新技術文章、程式展示及相關範例程式碼等等。

BEA dev2dev 是一個完整開發設計好的網站，目的是確保開發人員都能藉由 BEA WebLogic Workshop 8.1™獲得開發程式的高生產力。

BEA dev2dev 針對不同層級的開發人員提供許多珍貴的專家技術資訊、開發工具、訓練手冊等支援資源。BEA dev2dev 網站提供單一入口讓開發人員下載 BEA WebLogic Workshop™軟體、相關技術文章及白皮書，也提供相關的新聞群組郵件、訓練課程、研討會以及程式碼下載等功能。

其它重要的開發資源如下：

- BEA 技術支援: http://www.bea.com/support/
- BEA 教育訓練: http://www.bea.com/education/index.shtml
- BEA 在台灣的教育訓練: http://www.bea.com.tw
- BEA 開發者新聞群組: http://www.bea.com/support/newsgroup.shtml
- BEA 開發人員認證: http://education.bea.com/training/CertificationProgram.jsp

關於比爾亞(BEA)

BEA Systems比爾亞系統（Nasdaq: BEAS）是全球領先的應用基礎結構軟體供應商，爲全球超過15,000 個客戶提供企業軟體基礎架構，其中大多數客戶爲財星雜誌全球五百大企業 (Fortune Global 500)。BEA WebLogic®所提供的功能，能夠簡化應用程式的流程、降低應用程式的管理成本，並使得整個企業程式更輕巧，更具有競爭力。

同時，BEA 的平臺也提供超過 1,600 個系統整合商、軟體零售商(ISVs)、及應用伺服器供應商(ASPs) 完全解決方案，以確保其企業顧客快速成長及高競爭力。

BEA總部位於美國加州聖荷西市，在全球 31 個國家共有 77 個辦事處。更多BEA相關資訊，可瀏覽BEA網站http://www.bea.com / www.bea.com.tw

###